(4)

Technical Report 1220
July 1988

Machine Learning of Parameter Control Doctrine for Sensor and Communication Systems

R. B. Kamen
R. A. Dillard

NAVAL OCEAN SYSTEMS CENTER

San Diego, California 92152–5000

E. G. SCHWEIZER, CAPT, USN
Commander

R. M. HILLYER
Technical Director

ADMINISTRATIVE INFORMATION

This work was performed for the Office of Chief of Naval Research, Arlington, VA 22217–5000. The work was carried out by the Architecture and Applied Research Branch (Code 421) and the Artificial Intelligence Branch (Code 444) of the Naval Ocean Systems Center, San Diego, CA 92152–5000.

Released by
V.J. Monteleon, Head
Architecture and Applied Research Branch

Under authority of
J.A. Salzmann, Jr., Head
Information Systems Division

D.C. Eddington, Head
Artificial Intelligence Technology Branch

W.T. Rassmussen, Head
Advanced C^2 Technologies Division

PK

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	Approved for public release; distribution is unlimited.
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NOSC TR 1220	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Ocean Systems Center 6b. OFFICE SYMBOL (if applicable) Code 421/444	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) San Diego, California 92152-5000	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Office of Chief of Naval Research 8b. OFFICE SYMBOL (if applicable) OCNR-10P	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code) Arlington, VA 22217-5000	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO.: 61152N PROJECT NO.: ZT36 TASK NO.: AGENCY ACCESSION NO.: DN305 033

11. TITLE (include Security Classification)

MACHINE LEARNING OF PARAMETER CONTROL DOCTRINE FOR SENSOR AND COMMUNICATION SYSTEMS

12. PERSONAL AUTHOR(S)

R.B. Kamen, R.A. Dillard

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM TO	July 1988	108

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	artificial intelligence air-search radar agile beam

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Artificial intelligence approaches to learning were reviewed for their potential contributions to the construction of a system to learn parameter control doctrine. Separate learning tasks were isolated and several levels of related problems were distinguished. Formulas for providing the learning system with measures of its performance were derived for four kinds of targets.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL R.B. Kamen	22b. TELEPHONE (include Area Code) (619) 553-4007	22c. OFFICE SYMBOL Code 421

DD FORM 1473, 84 JAN

83 APR EDITION MAY BE USED UNTIL EXHAUSTED
ALL OTHER EDITIONS ARE OBSOLETE

CONTENTS

FIGURES

EXECUTIVE SUMMARY

OBJECTIVE

Increasingly complex sensor and communication systems are being developed and, as this complexity increases, the problem of analytically determining the optimum set of rules or control logic for these systems becomes computationally intractable. For this reason, a learning system is needed.

APPROACH

Artificial intelligence (AI) approaches to learning were reviewed for their potential contributions to the construction of a system to learn parameter control doctrine (in this case, a set of rules which can be used to control sensor or communication systems). Separate learning tasks were isolated, and the advantages, disadvantages, and limitations of the AI learning approaches relative to those tasks were determined.

Several levels of related problems were distinguished, ranging from the choice of parameters in a given situation to the design of a system of sensors capable of handling a given set of scenarios. In order to relate ideas on learning techniques to an application and to uncover some of the many aspects of the problem, a simple agile–beam air–search radar optimization case was chosen. Formulas for providing the learning system with measures of its performance (e.g., relative to target acquisition and track quality) were derived for four kinds of targets.

RESULTS

A study of the relative merits of different representations led to the selection of a production system architecture capable of learning control doctrine, and to the conviction that functionally independent aspects must be represented and accessed separately. A tool for constructing production system architectures from basic building blocks, PRISM, was obtained to facilitate experimentation with architectures for the learning system. One of the learning systems that resulted from this effort supports the following four different goals: (1) if the goal is to satisfy the constraints on the problem, a parameter setting is produced directly from the production memory that is augmented during learning to improve its ability to pick a near-optimum setting on the first attempt; (2) if the goal is to obtain an optimum solution, a series of solutions are reported, with each subsequent solution an improvement over the previous one, until the user stops the system from seeking a better solution; (3) if the goal is to learn, new productions are generated by one production memory to augment those in another; and (4) if the goal is to monitor the program's efforts, every step in the decision of a choice of parameters is reported.

INTRODUCTION

OBJECTIVES

The objectives of this research are both theoretical and practical. On the theoretical side, we seek to codify and extend the basic theory and techniques of artificial intelligence (AI) to encompass the problems which arise in domains such as radar parameter control involving numeric attributes embedded in a more complex conditional doctrine framework. We expect our orientation towards developing machine learning approaches to advance the development of techniques for the acquisition of knowledge which is not readily formalized, and to contribute to the development of tools for refining weak general problem–solving methods into strong domain specific ones. On the practical side, we hope follow–on work can carry the results of this research into eventually producing understandable and explainable collections of doctrinal rules for controlling the parameters of specific sensor and communication systems.

WHY MACHINE LEARNING?

Why pursue machine learning techniques for developing parameter control doctrine?

Analytical methods are intractable because of space or time on complex problems. For highly complex systems there are too many options and dimensions, and the computations are too time consuming. Heuristics for improving parameter settings may provide solutions satisfying constraints in cases where we were unable to obtain solutions before, and solutions of higher quality. The techniques may help to optimize some of the Navy's command, control, and surveillance functions.

An example of a typical constraint on a solution is the rule to reject any parameter set that results in a component performance measure out of range of its allowed values (Figure 1). (In our case, a parameter set will be rejected if it exceeds its maximum allowed value since the measures have been converted to range from 0 on up and each has a specified maximum allowed.)

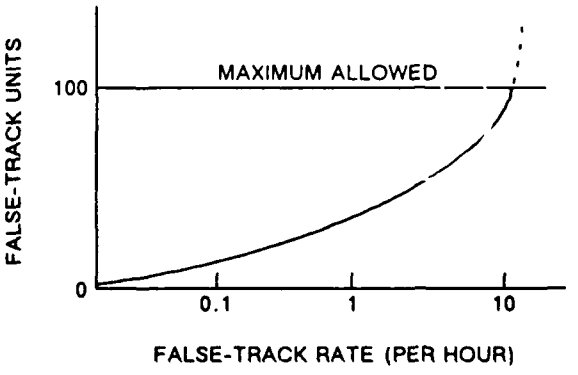

Figure 1. Example of a constraint on a component performance measure – a limit on false-track units.

LEVELS OF PROBLEMS

There are three levels of problems. In the first level, a particular sensor or communication system is functioning in a specific situation, and the problem is to optimize performance by appropriately setting the relevant parameters. Examples of parameters which may be set are the direction, power, and frequency for a sensor beam.

In the second level, instead of a single situation, the system is matched to a set of situations/scenarios and must find a control logic appropriate for all the situations and scenarios.

In the third level, a system of sensors or communication elements must be designed to deal with expected scenarios. If, instead of a single system, the designed system is a complex multiple system, it will have to distribute tasks among receivers in addition to setting the parameters for each receiver.

To summarize, the levels of problems are as follows:

(1) Particular optimization problem

Given: A particular sensor or communication system, fully described (its specific capabilities identified)

Situation and environment

Requirements on a solution

Find: (three sublevels of increasing difficulty)

(a) A set of parameters which satisfy the requirements on a solution;

(b) A near optimum set of parameter settings; and

(c) The set of parameter settings which optimize the performance of the system.

(2) Learning of doctrine for parameter control

Given: A fully described sensor or communication system

A set of situations and scenarios it is likely to face

Find: Control logic to make the system function optimally or near optimally, whatever the situation or scenario.

(3) Design of a system of sensors, given scenarios

(a) A single system

(b) A complex multiple system, such as an intercept system consisting of several receivers for warning, analysis, and direction finding, where it is necessary to divide the effort between components of the system as well as to set the parameters of each component.

THE LEARNING TASK

BASIC REQUIREMENTS FOR LEARNING

The learning system must be able to do the following:

Generate alternatives — produce new behaviors from which to learn from its "mistakes."

Determine the results when performance is

satisfactory

unsatisfactory

better than prior performance \ for a system attempting to

worse than prior performance / optimize performance.

Attribute credit or blame to some component of the performance system.

Modify its future behavior.

DECOMPOSITION OPTIONS

In tackling complex applications, it is advantageous to decompose the problems to devise shortcuts for finding their solutions. Five broad options for decomposition are assessed for the problem or learning control doctrine.

The first option is to break up the effort by parameter. After generating and solving a set of subproblems for learning to set (or change) each parameter separately, the resultant doctrine for a given parameter can be assembled with collections of rules for other parameters to form the complete set of doctrinal rules. Once the doctrine is learned, rules for different parameters can be collapsed wherever they have the same antecedents (left sides), and the consequents (right sides) can be compounded. Although this alternative sounds simple, it does have some disadvantages. Parameters may not be entirely independent. Thus, the learned doctrine for a given parameter could be affected by how the other parameters are assumed to be assigned. Performance measures are a function of several parameters at once, and optimizing those measures involves the simultaneous modification of several parameter values.

A second way to decompose the learning problem is to break it up into a set of subproblems for applying optimization techniques individually, that is for finding conditions on the application of the optimization techniques. Again, the resulting collections of doctrine may be assembled and integrated. However, this decomposition option is only appropriate for Task F (page 5).

A third approach is to break up the learning problem into a set of sub-problems, one for each performance measure. This will prevent us from overlooking valuable factors which partially cancel out when all performance measures are combined into the overall measure. We are pursuing this approach.

A fourth approach maps problems into simple representations. Solutions to simpler problems can function as skeletal solutions to more complex problems. Information gleaned in solving simpler problems can be applied to more complex problems. Additionally, a repertoire of techniques for handling simpler problems can be constructed along with the guidelines for combining them when tackling more complex problems. And, finally, we may choose the most appropriate representations of the problems for both learning and performance, even if they differ radically, and set up mechanisms to translate between the representations. In this case, the translation of learned material can be interpreted as operationalizing that information.

A fifth option is to construct intermediate features which classify the situation with respect to the appropriate techniques for obtaining performance goals. The intermediate features could be either features of parameters or features of conditions (situations). This move introduces new levels of abstraction in which the number of dimensions to consider is reduced. In some domains, dependencies between factors can be found. Then it becomes possible to extract primitives from which some dependent factors can be derived, enabling simplification of representations at another level. A technique called proportionality graph analysis is used to assess the contribution of this approach.

Additional strategies, which do not fit so neatly under the designation of decomposition options, such as analyzing extremes, exploiting symmetries, perturbation analysis, and finding analogies to solved problems are also considered.

SEPARATION OF LEARNING TASKS

Nine learning tasks have been isolated from the domain, and designated as Task A, Task B,... and Task I. What must be learned for each task is as follows:

Task A: Principles of assigning values to parameters.

Task B: Principles of modifying parameter values.

Task C: Intermediate features of conditions describing situations (characteristics of situations that occur together and that have the same significance for parameter control decisions).

Task D: Intermediate features of parameters (settings or changes of values to be made together).

Task E: Equivalent units to use in balancing different performance measures.

Task F: Heuristics to use in applying performance optimization techniques.

Task G: Empirically discoverable quantitative relationships between performance measures and parameter values.

Task H: Principles for diagnosing a situation as one of a type requiring a specific approach or treatment.

Task I: Paradigm situations to draw upon via analogy to determine parameter settings or changes.

This project addresses Tasks A through D.

COMPONENTS OF EMPIRICAL LEARNING

Learning from experience involves the following four subproblems: aggregation, clustering, characterization, and storage or indexing.

The aggregation problem consists of identifying the objects and their attributes (or entities) from which concepts will be formed. These are the objects or entities on which learning will take place. To solve this problem the learner must identify part–of relations.

The clustering problem consists of identifying which objects, entities, or events should be grouped together into a class to generate an extensional definition of the concept to be learned. The learner must identify instance–of relations to solve this problem.

The characterization problem consists of formulating some general description or hypothesis that characterizes the instances of the concept. The learner solves this problem by generating an intensional definition of the concept.

The storage or indexing problem arises for learners that need to use the knowledge they acquire. The characterization generated in solving the last problem must be stored so that it may be retrieved and employed in the performance of tasks.

SETS OF RULES

Performance Set

Conditions are descriptions of the situation.

Actions are suggested parameter settings.

This set is the parameter control rules system.

Learning Set

Conditions are descriptions of parameter setting cases and their associated performances.

Actions consist of adding rules to the performance rule set or altering the weights of rules already part of the set.

THE RULE LEARNER

The rule learner is a collection of mechanisms for the following:

generating new rules (often variants of existing rules);

weighting rules. (If they are relearned or used advantageously, they may be strengthened.);

abstracting to more general rules;

discriminating poor rules. (Making more specific variants in response to finding differences between contexts in which a given rule suggests less valuable actions, and contexts in which this rule suggests more valuable actions.);

proceduralizing memory elements to form rules;

composing rules into a single rule where they fire in sequence; and

open–ended possibilities for the construction of arbitrary LISP (or other) functions to encode additional techniques as needed.

APPROACHES TO TASKS

Solutions to the Aggregation Problem

There are several ways to look at the learning tasks. For example, various solutions to the aggregation problem can be seen as differentiating Task A through Task D and Task F (page 5). If individual cases or instances are chosen as the entities on which learning will occur, we will have a collection of cases each consisting of a set of values for situation variables and already set parameter values. Among the values could be included the performance measures taken of results of the setting event. Thus, we would have the right ingredients for learning intermediate features of conditions describing situations as in Task C.

If, instead of individual cases, entities become pairs of cases combined with the associated information on performance measures, we can accomplish Task A (learning the principles for assigning values to parameters, given the situation). Individual cases of changes in parameter settings form the entities for Task D enabling us to learn the intermediate features of parameter changes to be made together. Pairs of cases of changes in parameter settings allow us to tackle Task B and learn the principles for modifying parameter values. The entities needed for Task F are traces of the application of a curve–fitting or other numerical technique to a domain optimization problem.

Solutions to the Clustering Problem

The solution to the Task A clustering problem involves assembling the pairs on the basis of whether the parameter settings meet the constraints on solutions, and on the basis of their ratings on the performance measures. There will be a minimum of four classes of pairs: (1) a class to place pairs if both fail to meet the constraints

on a solution; (2) a class to encompass pairs of which one is a viable solution and the other is a failure to meet the constraints on a solution; (3) a class to place pairs of instances where both are solutions, but one has a better overall performance measure than the other; and (4) a class of pairs of solutions each having equivalent ratings on the overall performance measure. Using the relationship of the performance measures for each of the two cases in the pair, we can examine two classes for each performance measure, one instance as better vs. both equivalent in performance. However, these will be overlapping classes and will need to be handled in special ways. There is also the option of using the balance of the performance measures in clustering the pairs.

For Task B, the pair clustering results in three classes. Both parameter changes may show an improvement in performance measures, both may be degraded, or one may be improved and one degraded. Finer distinctions may be made on the relative performance measure changes in each instance, on which performance measures are involved and how they are improved or not in efforts to achieve balance.

The solutions to the clustering problem for Tasks C and D also use the performance measures, and even more gradations are possible in the approach to these tasks. The performance measures can extensionally identify a minimum of two classes above and below a threshold value for the performance measure where we select a single cutoff, or we may use multiple cutoffs for the desired number of classes. After characterizing the clusters symbolically using observable attributes, the process of clustering and characterizing continues by generating subclassifications of the clusters for each parameter setting.

Solutions to the Characterization Problem

The concept to be learned for Task A is a complex disjunctive relational concept, "<parameter setting> is good for <situation>". This 'is good for' concept can be interpreted in multiple ways as expressing quality for an overall performance measure or for each separate performance measure, thereby heralding a series of concepts.

Likewise for Task B, the concept to be learned is "<parameter setting> improves <performance measure> in <situation>". If the overall performance measure is assumed, <performance measure> need not be explicit in the relation.

Initial rules for setting parameters can be treated as hypotheses and used to predict whether an instance will be positive or negative. Such predictions can be divided into four classes:

A description can correctly match against a positive instance of the concept being learned. Using Task B for the example, a parameter change which leads to an improvement in performance can be seen as a positive case of the rule which proposes it.

A description can correctly fail to match against a negative instance of the concept being learned. None of the rules should match cases in which a parameter change leads to a degradation of performance.

A description can incorrectly fail to match a positive instance of the concept. This is called an error of omission, and suggests that the hypothesis or rule is too specific.

A description can incorrectly match against a negative instance of the concept. This is called an error of commission, and suggests that the hypothesis or rule is too general. The rules should not propose parameter changes that degrade performance.

Solutions to the Storage or Indexing Problem

The representation of acquired knowledge for Tasks A, B, and F will reflect the need for the use of that knowledge in the production system architecture. On the macrolevel, this information is stored in disjunctive rule choices, while on the micro–level, it is entered as production rules. We implement rules which themselves have action–sides specifying that additions and changes be made to certain procedural memories. Some of the learned material is hidden in alterations to activation values or strengths associated with already present rules. This will be discussed in more detail under the description of the PRISM architecture building system.

For Tasks C and D, concepts are constructed which may be added both to object hierarchies and to the individual cases as new attribute–value pairs. At present, the additions are not made automatically during the functioning of the learner as they are for Tasks A and B, but must be done by hand in setting up the representations for the instances and in constructing the hierarchies.

There are a variety of alternatives for indexing and recognizing concepts stored hierarchically. One of the best approaches appears to be Gluck and Corter's (1985) category utility which measures the average extent to which knowledge of category membership increases certainty about the attribute values of category members. Formally, given a partition {C1, C2,...,Cn} over a set of objects which have attributes Ai with possible values Vij, category utility equals:

$$\left[\sum_{k=1}^{n} P\left(C_k\right) \sum_i \sum_j P\left(A_i = V_{ij} \middle| C_k\right)^2 - \sum_i \sum_j P\left(A_i = V_{ij}\right)^2 \right] / N$$

This value is used as a measure of the quality of concepts in an incremental concept formation system, COBWEB/1 (Fisher, 1986).

DESIRED FORM OF LEARNED DOCTRINE

Since we wish to generate doctrine for use by humans, we have been most attracted to doctrine as expressed in rules. Learning systems can very naturally be expressed using rule–based systems.

Overview of Production Systems

In terms of structure, production systems consist of two main components:

> (1) a dynamic working memory containing the current state of knowledge in the problem solving effort; and (2) a more stable production memory, stated as a set of condition–action rules or productions;

In terms of behavior, production systems have three main processing stages:

> (1) the match process, in which conditions of rules are matched against the contents of working memory to determine what rules will be applicable to the current problem; (2) the conflict resolution process, in which some rules are selected for application; and (3) the act process, in which selected rules are executed.

Production systems operate in cycles, with actions on one cycle leading to new matches on the next cycle because of changes in the working memory.

Advantages of production systems for learning are the following:

> Homogeneity – all rules have similar form.
>
> Independence – knowledge is broken up into small modules.
>
> Stimulus/response flavor – conditions may 'respond' to external stimuli.
>
> Goal–driven behavior – conditions may be sensitive to goals.
>
> Parallel or serial in nature – recognize–act cycle.
>
> Separable memories – blackboard and knowledge base are analogous to short– and long–term memories.
>
> Rule preference can change over time using activation or strength–based conflict resolution.
>
> Modularity of knowledge makes it possible to learn by adding productions in addition to strengthening productions.
>
> Variants can be derived by generalization (specific to general); discrimination (general to specific); composition (combining rules); and proceduralization (instantiating rules).

THE PRISM ARCHITECTURE BUILDING TOOL

PRISM (Ohlsson & Langley, 1986) provides the building blocks for synthesizing production system languages. We use PRISM to formulate a system capable of learning heuristics for controlling the parameters of sensor and communication systems.

PRISM describes a very comprehensive space of interesting production system architectures by providing ways to specify the organization in terms of components (or schemas), each of which has a small set of associated parameters. The PRISM building blocks may be used to define numerous well-known production systems, such as OPS4 and ACTE, in addition to others generated for individual research projects or theses.

The developers of PRISM consider its major contribution to be that it constitutes a framework for the discussion and comparison of alternative production system architectures. For this project, PRISM's contributions were that it facilitated experimentation with architectures for learning, and it eliminated some of the routine programming implementing new production system languages.

Except for the rete-network pattern-matching process developed by Forgy (1979), which was chosen for its efficiency and the overall structure of the interpreter, nearly every aspect of the default PRISM architecture can be modified by the user. The rete pattern matcher will handle virtually any structure. Notations chosen for representing knowledge are neither legislated nor enforced by the PRISM architecture. We can define as many working memories and procedural memories as needed, and may create selection processes to execute those productions (found by the matching process to be satisfied) on the basis of the cycle or the order in which the element was added to working memory, complexity, or activation. Each declarative memory can have quantitative attributes which we may interpret as activation, strength, probability, etc., and which may be initialized, propagated, updated, accessed, and used differently for different memories. Different procedural memories may be given different selection schemes and be organized hierarchically.

The flexibility of PRISM allows productions in one procedural memory to generate productions that can be added to another. In addition, all kinds of experimental organizations of productions can be pursued simultaneously.

RULE STRUCTURING AND ORGANIZING

Metarules can be generated automatically to partition the domain into manageable segments. The value of adding a particular metarule to the rule collection can be calculated by examining the cost of evaluating that rule (a function of the number of conjuncts in its premise) with respect to the product of the chance of the premise being evaluated to true and the amount of pruning of the space of rules accomplished by the metarule. These metarules could focus attention on the segment of the knowledge base having rules with premises capable of evaluating to "true" given the current facts. If a particular conjunct appears in many different rule premises, evaluating this conjunct first and ordering the rules on the basis of this

conjunct could improve the efficiency of traversing the rule space. Metarules generated to effect this ordering could take the form of ruling in a collection of rules, or of ruling out some collection of rules, or both.

Metarules could distribute productions to different procedural memories. In PRISM, the rule space can be partitioned and the resulting partitions treated as if they were never segregated, or in a whole panorama of different ways.

LEARNING TO LEARN

A metalevel of learning can be described in which the approach is to learn heuristics for searching the rule space. The learning literature contains various examples of learning heuristics to direct search through problem spaces. The techniques used in these learning systems can be applied to the problem of learning heuristics to direct search through a space of rules. Thus, a system should be able to start with very general and simple learning methods and improve its ability to learn with experience. All that is needed is a complete solution path to use as the basis for assigning credit and blame. Successive hypotheses form a path through the space of rules. If the techniques for generating the path of hypotheses can be described, then we can assign credit or blame to those techniques which created or learned the rules used in the performance system. Once these techniques are isolated, the approach can be applied recursively, essentially collapsing any rules for learning to learn to learn into the first metalevel of learning to learn. However, no obvious way exists to represent powerful methods, such as data–driven generalization or discrimination techniques, in terms of modifiable rules at the metalevel. For the learning of learning heuristics, it is necessary to represent those techniques and all conceivable variants so that they may be "found" by the learner learning to learn.

However, in the particular domain of radar parameter doctrine learning there may be a way to specify some of the metalevel learning principles. This is an area for future attention.

AI APPROACHES TO LEARNING

GENERALIZATION

One method for learning symbolic conditions begins with very specific information and generalizes as more data become available. Usually the hypothesized conditions are initialized to the first positive instance. A new positive instance is compared to the current hypothesis and only those features held in common by the instance and the hypothesis are retained in one or more revised hypotheses. If some hypotheses become too general, incorrect classifications occur and the hypotheses will be discarded.

As a technique, generalization refers to a method for abstracting from the details of positive and negative examples to form hypotheses (rules) which cover the data. This technique cannot recover from learning trials in which instances are misclassified, so it is unsuited to noisy data. It also requires that the concept to be learned does not change over time, and that the representation language is sufficient to describe the concept. Generalization has difficulty learning rules with disjunctive conditions, since the search is for features held in common by all of the positive instances.

Descriptions can be made more general by removing conditions, replacing constants with variables, replacing a term with another which is its ancestor in a tree expressing what are called IS–A hierarchies, adding elements to lists of allowable elements, or increasing the size of an interval of allowable values. Work using generalization techniques is described by Winston (1975), Vere (1980), Mitchell (1982), and Dietterich and Michalski (1983).

Generalization–based approaches to learning are not suited to the learning of heuristics because they do not provide sufficient variety. In learning heuristics, the system begins with extremely specific hypotheses which are only capable of very conservative moves, so it will begin by making no bad moves and missing some good moves. Where a system must generate its own instances of the use of heuristics, there is no great penalty for errors of commission, which indeed are necessary if the system is to learn, and yet with the generalization method, very few will be made. However, the price of omissions is great since learning is impossible if the behavior which provides the instances is absent. Generalization is most appropriate if there is a benevolent tutor to present positive and negative examples which are typical and which cover the space of the concept's applicability.

However, for heuristics learning, analytical generalization (in contrast to simple generalization) is highly attractive because of its minimal need for examples. Each case is milked fully by attending to the reasons (causes) that make it an instance of the concept of interest. This focuses the generalization effort toward just those features which are important to the goals at hand. The literature contains programs which are able to learn new concepts on the basis of just one instance. The background knowledge of causal connections may have to be considerable, however.

Analytical generalization, which will be discussed further below, would be ideal as an adjunct to constructive techniques which can add the variety missing in simple generalization techniques. Constructive techniques free the learning system from the dangers of representations which restrict the forms that can be learned too severely. The bias of the learning system is flexible in systems with constructive capabilities.

DISCRIMINATION LEARNING

This learning technique starts from one or more overly general rules and generates more specific versions through a process of discrimination. This occurs when one of the current hypotheses (rules) leads to an error. The context in which the faulty rule matches the negative instance is compared to some context (usually the last context) in which that same rule matches a positive instance. The comparison reveals differences between the positive and negative instances. For each difference found, a more specific hypothesis can be constructed (or strengthened) which will match against the positive and not the negative instance. The ability of each hypothesis to account for the data distinguishes useful variants of hypotheses from spurious ones.

Descriptions can be made less general by adding conditions, replacing variables by constants, replacing a term with another which is its descendant in a tree expressing what are called IS–A hierarchies, removing elements from lists of allowable elements, or decreasing the size of an interval of allowable values. These are just the reverse of steps described above for generalization.

The discrimination learning technique can handle both relational and attribute–value representations, it can acquire rules containing complex negated conditions, it can learn from far misses as well as near misses, it can learn in the presence of considerable noise, and it can learn rules which have value heuristically in spite of inadequate representations. Since the discrimination method compares instances to other instances rather than to hypotheses which summarize those instances, it does not attempt to find features common to all positive instances, and can easily learn rules with disjunctive conditions. We use this learning method in our research. Other work with the discrimination method includes ID3 (Quinlan, 1983), and SAGE (Langley, 1985).

The discrimination method is ideal for the learning of heuristics because, in the move from general to more specific hypotheses, discrimination will not at first omit desirable moves, although it will consider many undesirable ones as well. The performance will become less rash as the correct hypothesis is approached during the discrimination process. Only those heuristics which have been proven to be inappropriate will be removed from consideration.

THE VERSION SPACES METHOD

The version space approach incorporates aspects of both the generalization and discrimination methods (Mitchell, 1977). This approach begins with a very specific hypothesis and generates more general versions (S) by extracting common features of the newly encountered positive instances. It simultaneously begins with a very general hypothesis and generates more specific versions (G) as more data is processed. However, instead of testing the first set of hypotheses (S) against negative instances to see if they are overly general, it tests them against the already processed set (G). Similarly, more specific versions of the set (G) are found by comparing negative instances to the hypotheses in the processed set (S). As more instances are gathered, this multidirectional process converges on the hypothesis best suited to summarize the data.

Unfortunately, this method encounters almost all of the disadvantages of the generalization approach. It searches for features held in common by all positive instances, so it has the same difficulty with disjunctive rules, noise, inadequate representations, and concepts that change over time. The only problem it does not have is the introduction of variety, which is accomplished by the discrimination component of the technique. We are not currently employing this approach in our research.

EXPLANATION-BASED GENERALIZATION

Explanation-based generalization, also called learning by analytical generalization, starts from a goal concept, a training example, a domain, and an operationality criterion. An operational description of the goal concept that covers the training example needs to be found. More precisely,

goal concept = a nonoperational definition of the concept to be learned.

training example = a single positive instance of the goal concept.

domain theory = a set of rules to be used in explaining how an entity satisfies the goal concept.

operationality criterion = a test specifying the form of the definition to be learned.

The process consists of two steps, explanation and characterization. The explanation is constructed using the domain theory to prove that the training example is a positive instance of the goal concept. All the nodes of the explanation tree produced by this process must be operational. The characterization is constructed by determining a set of sufficient conditions under which this explanation holds, stated in operational terms. The goal concept may be regressed through the explanation tree to accomplish this.

This analytic approach to learning has various advantages. It does not require extensive search through the space of concept descriptions or rules, since the explanation leads directly to the learned description. It handles disjunctive concepts, since it finds only sufficient (rather than necessary) conditions on the concept. It can learn from a single positive training instance; negative instances are not required. It can handle noisy data, since the explanation process will catch misclassified instances. Finally, it provides a justification for the characterization of the concept description which is based in the operationality criterion and the explanation.

There are also some disadvantages to the analytic approach based on explanation. It lacks generality since it must be grounded in a rich enough framework of domain knowledge to form the explanations. The search that is saved in the space of concept descriptions is required in the space of explanations. Where multiple explanations are appropriate, search through the space of concept descriptions may still be necessary.

Within the domain of radar parameter control, there is a body of analytic relationships which may lend themselves to exploitation through explanation–based learning. This is an area for future attention. Researchers in this area include Mitchell, Keller, and Kedar–Cabelli (1986), and DeJong and Mooney (1986).

LEARNING AS SEARCH

Search plays an important role in machine learning, as it does in the rest of artifical intelligence. In machine learning, search occurs in the space of rules or hypotheses rather than just through the space of simple problem states. Most machine learning systems use rather straightforward search methods (such as depth–first or breadth–first search), and search through the space of hypotheses often takes advantage of its partial ordering on the basis of the generality of the hypotheses. However, the operators for search may themselves involve search, and are generally more powerful than operators found in other problems for which AI approaches are used. The direction of search we saw could be from specific to general, as in generalization, or general to specific, as in discrimination, or both, as in the version spaces method.

Search always occurs in some problem space, and can be constrained in a variety of ways. The constraints on search in the context of machine learning are called the bias of the learning system. Bias can be attributable to the representation or to the search organization. Representational bias results from considering only certain features of the instances (only those expressed in the system), or from allowing only certain forms of hypotheses. Many learning systems are restricted to learning only hypotheses which can be expressed as conjuncts, for example.

Search bias occurs when hypotheses that violate certain constraints are eliminated from consideration. For example, many systems will reject any hypotheses which are inconsistent with the data, thereby making them sensitive to noise. Ordering hypotheses by some criterion also constitutes a bias. In Appendix B, we list the preference criteria considered for inclusion in our learning system. All learning systems must have biases or no learning could take place. Methods for shifting or

weakening inappropriate biases may include changing the characteristics of the language in which hypotheses and data are described, altering the space of hypotheses which may be considered, varying the procedures which define the order in which hypotheses are considered or valued, and changing the acceptance criteria for determining when a search procedure has found a "good enough" candidate hypothesis.

Strong bias is one that focuses the learner on relatively few hypotheses. An incorrect bias is one which prevents the learner from discovering the correct hypothesis. Paul Utgoff (Utgoff 1984) addresses the problem of altering incorrect or weak biases to improve learning abilities.

CONCEPTUAL CLUSTERING

Conceptual clustering is an outgrowth of the older data–analysis methods of cluster analysis and numerical taxonomy in which the task is to find a hierarchical classification tree that summarizes data given in the form of attribute–value representations of a set of objects. The intent is to maximize intracluster similarity while minimizing intercluster similarity. Distance in N–space is commonly used to measure similarity. However, only the objects themselves are used to evaluate the clusterings and, in the worst cases, the results can be uninterpretable.

Michalski and Stepp (1983) have formulated the task of conceptual clustering which, in addition to generating the hierarchy, demands that intensional descriptions of the resulting clusters be produced. The quality of those descriptions figure in the evaluation of the clusters. The resulting clusters are more conceptually coherent than those generated by traditional methods.

In numerical taxonomy approaches, objects are represented as points in an N–dimensional space. The method typically proceeds by creating a cluster from the two closest objects in the space, removing them and creating a new object whose coordinates are the average of its members' coordinates, continuing until all objects are included in a single cluster.

In contrast, CLUSTER/2 (Michalski & Stepp, 1983) employs generalization to cover all the objects from an initial selection of N seed objects, using them to simulate positive and negative instances. Once the set of objects is clustered on the first round, a new set of N seed objects which constitute the central tendency of the resulting clusters is chosen, and the process of generalizing to cover the remaining objects is repeated, until the system stabilizes on a particular set of seeds. Finally, each of the clustered sets of objects is characterized, producing a description for each cluster. The entire method is repeated for different numbers of seed objects (where N ranges from 2 to about 7), and the best clustering, according to the criteria for judging the descriptions, is used to add branches to the tree which will embody the classification. The process is then applied recursively (and separately) to each of the resulting clusters until no additional progress is made by further subdivisions.

A simpler conceptual clustering system, RUMMAGE (Fisher, 1984), is model driven and uses knowledge of attributes to divide objects into groups. The attribute that divides the objects so that their resulting characterizations are the best is used to form the branches of the tree at each point.

Conceptual clustering involves three levels of search, the search for clusters of objects, the search for characterizations of clusters, and the search for hierarchies of the clusters. Our clustering program uses a model-driven approach to find the clusters of objects and, like RUMMAGE, uses the attributes individually to specify branches at each point in the hierarchy. However, it is simpler than RUMMAGE since it uses as its criteria of quality the aim of minimizing the total number of nodes in the tree. This is not an exact match to value for a classification tree, since number of nodes is not coextensive with ease of access, or even with minimum number of tests to make a decision using the tree. However, it does provide an initial cut for clustering objects in a new domain. Additional work on conceptual clustering is described by Fisher and Langley (1985 and 1986), and an incremental conceptual clustering is introduced in COBWEB (Fisher, 1986).

CHUNKING

Chunks were originally proposed to explain short-term memory phenomena (Miller, 1956). The term "chunk" refers to a familiar pattern that can be easily remembered and manipulated as a single entity. Chunks figure in cognitive psychology to account for the differences between novices and experts. They may be either perceptual in nature or action-oriented and can involve either spatial or sequential action structures. Most of the chunks appearing in AI involve sequential action structures.

In SOAR (Laird, Rosenbloom, and Newell, 1986), an architecture has been built for problem solving and learning by chunking. In the context of a means-ends analysis framework, the architecture creates subgoals as difficulties are encountered. The subgoal then creates a chunk stated as a production rule which allows it to bypass the need to generate and achieve that subgoal in the future.

The rule created by SOAR includes in the conditions for the rule all memory elements that were present when the subgoal was created which were subsequently matched during the processing of that subgoal. It includes on the action side of the rule all memory elements constructed during the processing of the subgoal which can be reached from the subgoal by chaining through links. All nonterminal symbols are replaced by variables, resulting in the formation of relatively general rules. SOAR does not search through a space of possible rules. The problem solving progress determines what rules are learned; learning is one-trial and automatic.

Chunking in SOAR leads to different effects depending on what difficulty is resolved by the subgoal. When the subgoal is to determine what operator to apply, SOAR learns heuristics; when the subgoal is to determine which state to select, SOAR learns an evaluation function on the states; and when the subgoal is to apply a complex operator, SOAR learns macro-operators represented as a set of rules. Different forms of learning in SOAR arise not from different learning mechanisms, but from variety in the problem solving process.

MACRO-OPERATORS

Machine-learning researchers have explored two approaches to improving problem solving performance: learning heuristic conditions on operators to constrain the search process, and defining higher- level operators which shortcut the steps needed to traverse the problem space. The latter approach is called learning macro-operators. Such "macros" can be represented either as a single data structure in which a list of operators is specified, or as conditions on independent operators which cause them to be applied in a certain order.

Problems which do not have nearly decomposable subgoals require the problem solver to temporarily violate goals which were achieved earlier. Macros can be formed for some of these processes to allow the problem to be tackled in a new problem space in which intermediate steps can be ignored. For problems expressible in terms of evaluation functions, macros can be seen as bypassing local valleys (or peaks) in order to achieve a solution or a more robust solution. This approach to learning is used and described by Neves and Anderson (1981), Korf (1982), Iba (1985), and Anderson (1983, 1986).

HEURISTICS LEARNING

The task of learning search heuristics can be formalized as follows: GIVEN: (1) an initial state from which to begin the search (for one or more problems); (2) a set of operators for generating new states along with their legal conditions for application; (3) a test to determine if the goal has been reached; and (4) a search strategy for selecting operators and states, FIND: Heuristic conditions for each operator which will reduce or eliminate search.

Methods for strategy learning can lead to improvement in which search is reduced on practice problems during their solution, or several varieties of transfer in which search is reduced on new problems. The transfer may be to scaled-up problems, to problems with different initial and goal states of the same complexity, or to problems also of the same complexity but with a differently structured state space for which learning by analogy (Carbonell, 1986) may be appropriate.

Learning heuristics has been accomplished by learning from solution paths (Langley, 1985). Once a solution path is found, credit and blame can be assigned to instances of the application of rules. Each move along the solution path can be marked as a positive instance of the rule that proposed it, and each move one step off of the path can be marked as a negative instance of any rule that proposed it. The positive and negative instances can then be passed to some characterization method to determine the heuristic conditions on each move-proposing rule.

Where it is not feasible to wait for a solution to begin the process of learning heuristics to constrain search, moves can be classified as undesirable during the search for solutions by noting loops, flagging dead ends, detecting illegal states or moves, noting failure to progress towards a goal, or detecting less redundant or shorter paths, as in the Universal Puzzle Learner (UPL) system (Ohlsson, 1983). Moves can also be classified as desirable if shorter and less redundant paths are recognized, or identifiable progress towards goals is noted.

LEARNING EVALUATION FUNCTIONS

A method for selecting states for further expansion during problem solving involves the construction of a numerical evaluation function to rank the states and, thereby, control search. A checker–playing program (Samuel, 1959) chooses moves on the basis of a linear evaluation function. The system experimentally introduced new terms from a set of predefined features, adjusted the weights of existing terms, and took note of the resulting effect on its playing ability.

In domains with numerical attributes, curve–fitting techniques can be used to generate functions which predict known ratings of states in terms of sets of predefined features, and the resulting functions can subsequently be used for directing the search process. Although these techniques can be valuable for domains rich in continuous attributes, they are not very well suited to acquire heuristics which can only be stated in symbolic terms. Such symbolic variables could be translated into a set of dichotomies which have either zero or one as possible values, but some information is lost pertaining to the constraints between specific values of the original nominal variable, and the evaluation function may end up being rather awkward to construct.

THE COMPOSITION MECHANISM

Composition operates by collapsing multiple productions into a single production which has the effect of executing the set of productions. For the productions to be collapsible, they must not contain actions which consist of arbitrary Lisp functions, that is, their actions sides must be transparent. Actions which delete or remove elements can also cause difficulty, as can negated conditions. In early research it was assumed that composition should occur whenever two rules were applied in sequence, but Anderson (1983) evokes composition only when goals are achieved.

PROCEDURALIZATION

Proceduralization builds into the productions information which formerly had to be retrieved from long–term memory. This process forms more specific variants of rules which are used so that in subsequent accesses of those specific variants of the rule, the domain–specific information retrieval step is bypassed and the match process finds all of the necessary information in the interpretive production which was constructed. In ACT (Anderson, 1983), the combination of composition and proceduralization has resulted in a robust learning system. Recently, Anderson (1986) has attempted to explain how generalization and discrimination effects can emerge from this combination.

GENETIC ALGORITHMS

Genetic algorithms are search algorithms based on the mechanics of natural genetics. To apply genetic algorithms to the task of learning, John Holland (1986) of the University of Michigan constructed a system consisting of three main elements: (1) a rule and message system, (2) an apportionment of credit system; and (3) a genetic algorithm.

Starting from a randomly generated state, the learning system learns string-rules called classifiers which match strings called messages. Messages derive either from previously activated classifiers or from environmental sensors. The effectiveness of each classifier is evaluated, and new rules (classifiers) are created using the innovative search mechanism of the genetic algorithm. The classifiers are then used to guide the system's behavior in the environment. The environment impinges on the system through its sensors, called detectors. This information is decoded to some standard message format and the result is placed on a message list along with the rest of the messages generated by the previous cycle. Messages on the message list may activate classifiers, rules in the classifier store. Activated rules may send messages to the message list for the next cycle. In addition, certain messages may call for external action through a number of action triggers called effectors. Thus, both internal and external data are used through the message system to guide behavior and the state of the system for the next cycle.

The relative value of candidate rules is determined by the simulation of a competitive service economy. The right to respond to relevant messages is given to the classifier with the highest "bid". The payment of the "bid" is made to the classifier which previously sent the message, the successful message sender. In this way, a chain of middlemen is formed from "manufacturer" (source message) to message "consumer" (environmental action and payoff). The competitive nature of the economy insures that the good rules survive and the less valuable ones die off.

Messages and classifiers have a very simple representation in terms of binary strings with a wildcard "#" value which matches either a "0" or a "1". Thus the alphabet consists of {0,1,#}. String rules expressed in this form play the role of chromosomes and provide the raw material for the generation of new string rules constructed from a structured, yet randomized, exchange of information among randomly mated pairs of rules.

A genetic algorithms approach can be applied to learning optimum parameter settings for sensor and communication systems and is, in fact, the approach used in a battle management systems control rule optimization project at the Naval Surface Weapons Center in Dahlgren, VA (Kuchinski, 1985).

We reserved this approach for examination last, if expected progress did not materialize from other approaches. We were reluctant to use it, not because it was inapplicable to the domain, but because results from our other learning research could contribute to cognitive theories of learning.

NEURAL MODELING AND CONNECTIONIST APPROACHES

Neural modeling and connectionist approaches to learning in complex domains are very attractive since they do not require the kinds of preliminary analysis of the domain necessary for other approaches. In this framework, knowledge is typically represented in terms of weights on the links in a 'neural network'. Learning takes place by making modifications to those weights. The approach makes very few assumptions and, although it operates more slowly than other methods, is more general. Speedup may occur since it has a straightforward adaptation to parallel

implementation. Indeed another name for these systems is parallel distributed processing (PDP) systems. In these systems, the representation and processing structure are identical. Multiple representations are linked by local cooperation and competition. Learning and adaptability are fundamental primitives, and handling multiple goals, multiple constraints, and conflicts is fundamental to the system.

These systems are extremely powerful and robust both for computations and for representations. They are fault tolerant, and degrade gracefully at the peripheries of their areas of competence. Solutions that meet the formal constraints on a solution can be reached quickly, and solutions can be based on incomplete, ambiguous, errorful, and even contradictory data in both facts and goals. Knowledge is distributed across processing elements, yet rule-governed behavior emerges in the absence of rules. The starting state, plus goal conditions, transform the system into the end state.

However, the end product of the learner is a system capable of generating the desired behavior, not a set of rules or an explicit representation of what has been learned. Thus, if we used this approach to learning control doctrine for sensor and communication systems, we would have a system which, in some sense, learned the doctrine, but we would have no way of knowing what that doctrine was except by watching it being 'applied' by the system during operation. Consequently, for our research, this approach was tabled.

Recent work in connectionist models of learning have been carried out by Anderson and Hinton (1981), Kohonen, Oja, and Lehtio (1981), Barto (1985) with his self-interested model, Rumelhart, Hinton, and Williams (1985) with their generalized delta rule, Fukushima (1982) with his "neocognitron," and Hampson and Volper (1986). More needs to be done to improve the learning rate before the connectionist framework will be applied routinely to complex problems. Steve Hampson of the University of California, Irvine has recently employed conditional probabilites to cut the learning rate by possibly a factor of 200 or more.

OTHER APPROACHES TO LEARNING

This report does not discuss classical estimation and control, supervised and unsupervised Bayesian learning theory, learning logics (nonmonotonic and inductive logics), pattern recognition, search theory, and optimization techniques. These promising areas of research were not included because our investigations were already so broad.

A RADAR EXAMPLE

The radar situation used as an example in this report is a simplification of the general problem described in Dillard (1986). The concepts involved and the description of a two–dimensional air radar case are listed below.

THE GENERAL PROBLEM

Environment

Track data (dymnamics, cross sections, ID's...)

Intelligence

Electronic countermeasures (ECM)

Reaction times

Terrain, Weather

Possible Variable Parameters

Next pointing angle(s)

Angular coverage (1 pencil beam, 1 fan beam, n simultaneous beam, or a combination)

Per Beam

PRF (pulse repetition frequency)

Dwell time

Waveform/resolution

Frequency

Power

Number of looks per scan

Per–Beam Target Situations (One or more)

Default

No track is likely to continue in that beam on that scan, and there are no indications from intelligence or electronic support measures (ESM) that a target is likely to enter from that direction.

Track continuation

A current track could continue in or enter that beam.

High speed

A target that could be in that beam is travelling at high speed.

Weak

Either small size or great distance is making the target that could be in that beam have a weak signal strength.

Close range

The target is likely to be too close for the use of a high resolution pulse.

Priority search

Intelligence, ESM, or other sensors have indicated a target might be entering from that direction any time.

Two in one bin

There is a likelihood of at least two targets in the same range bin in a given beam.

A TWO-DIMENSIONAL AIR RADAR EXAMPLE

Radar Description

Agile fan beam, 10-degree beamwidth

162-nmi instrumented range

300 pulses per second

Pulse duration = 100 μs (low resolution)

or 2 μs (high resolution)

Dwell time per beam = 0.1 second (30 pulses)

or 0.2 second (60 pulses)

Looks (beams) per beam position per scan = 1 or 2

Example of a parameter setting rule

(Situation Conditions -> Parameter Setting)

Situation Conditions

Low-density target environment

For the beam in question:

Priority Search

Not a track continuation

Expected target is not high speed

Parameter Setting

Dwell time = 0.2 sec

Number of looks (per scan) = 2

SIMPLIFICATIONS

Initially limit environmental considerations to track data and to warnings of potential targets.

Limit per-beam situations to just three categories:

Default

High speed

Other (All others are included here).

Assume all targets are radial, i.e., they remain in the same beam position.

Limit per-beam parameters to

Dwell time, with a choice of 0.1, 0.15 (an addition to the above example), or 0.2 second,

Looks per scan, with a choice of 1 or 2 looks.

(Omit considerations of low- vs high-resolution pulse duration.)

Consider four target types

High–speed (1000 knots) weak targets [to be called Target_W]

High–speed (1000 knots) average–strength targets [to be called Target_X]

Average–speed (400 knots) weak targets [to be called Target_Y]

Average–speed (400 knots) average–strength targets [to be called Target_Z]

(The term "weak" may be interpreted as conflating average–size targets at maximum range and small targets at midrange.)

Performance measures

False–alarm–rate units

Target–acquisition units for each of the four types of targets above.

Track quality units again for each of the four types of targets above.

Omit consideration of

Average–expended energy

Target resolution

Blind range

Limit attention to Task A in which the goal is to learn the principles of assigning values to the parameters, given the situation.

NOTATION

Appendix A describes this simplified problem in considerable detail. It includes examples of the situation space, parameter selection strategies, the parameter space, and component performance measures. A summary of the notation used is given below.

Let na, nb, and nc denote the number of beam positions believed to have situation a (default), situation b (high speed) and situation c (other), respectively.

Let a1, b1, and c1 denote the looks per scan to be used (next scan) in beams having situations a, b, and c, respectively.

Let a2, b2, and c2 denote the dwell time to be used (next scan) in beams having situations a, b, and c, respectively.

The simplest strategy calls for values of a1, a2, b1, b2, c1, and c2 to be selected for the next scan based on the current situation (na, nb, nc).

THE RULE LEARNING PROCESS

Sample Rules from the Performance Set

Most specific

If High–Speed & na = 35 & nb = 1

then b1 = 2 & b2 = 0.2.

If Default & na = 35 & nb = 1

then a2 = 0.1.

More general rules may also be added.

If High–Speed & na > 30

then b1 = 2 & b2 = 0.2.

As long as we start with a set of rules which spans the space of possible scenarios, we can start the learning process.

Input to the Learner:

At least two cases with associated details on

environment

sensor capabilities

parameters chosen

performance measures for that choice.

[History file of previous cases]

[Analytical relationships between variables for use in Explanation–Based Learning techniques]

Output from the learner:

Possibly – changes in weights associated with rules in the performance set

– additional rules to enter into the performance set
[removal of rules from the performance set]

Note: Removal of rules is not necessary from the point of view of improving the performance of the parameters chosen, since weights may be set so that a given no–longer–useful rule will not be used, but removal may be desirable to improve the efficiency with which rules are selected for execution during processing.

Allows incremental learning:

Any new case may be compared with all previous cases for modifying the performance rule set.

EXPERIMENTS

Initially, a conceptual clustering program was written to address the learning tasks of determining intermediate features of situations and parameters (Tasks C and D). This program uses a model–driven method for constructing nodes and minimizes the number of nodes in the clustering tree. It assumes that the variables provided for describing the situations and parameters are relevant to the task of choosing appropriate parameters.

The PRISM production–system architecture builder was used to construct an initial experimental performance production system for a two-dimensional agile-beam radar. This system proposes parameter settings given the situation and environment. Its behavior was to provide feedback to the learning system to begin the process of learning optimum choices. However, the performance system was built using an excessively complex model, and there was no assurance that the initial rules spanned the space of possible scenarios. Simplifications in the initial model warranted reconstruction of the performance system that was incorporated into this learning system. The pieces of the system responsible for performance can be isolated easily because they form separate production memories. The advantage of this integration with the learning system is that learned productions may be immediately put to use in solving problems.

The PRISM architecture builder was reimplemented to make it compatible with the LOOPS object–oriented language on the Xerox 1108. The intent was to make it possible for information housed in LOOPS hierarchies to be accessed by learning systems built using the PRISM architecture builder. However, making the two systems compatible resulted in a more complex syntax for PRISM. Additional familiarity with PRISM revealed that the advantages of hierarchical storage could be obtained within PRISM. Even for systems already implemented in LOOPS, transfer of all the material into linked PRISM declarative memories would be superior to losing PRISM's flexibility of associating numerical attributes to the elements in its memories for use in selection processes.

Details for a PRISM–based production system learner were worked out. This system consists of a set of rules for adding, modifying, and weighting rules in the performance system. Defining separate production memories in PRISM allows us to store performance productions in the best manner for functioning to set parameters, while storing learning productions in the best manner for learning.

The first step in our learning approach (to the agile–radar problem) was to investigate clustering, based on Task A. What follows is a description of a way of implementing this approach. Figure 2 gives the context in which experiments were made.

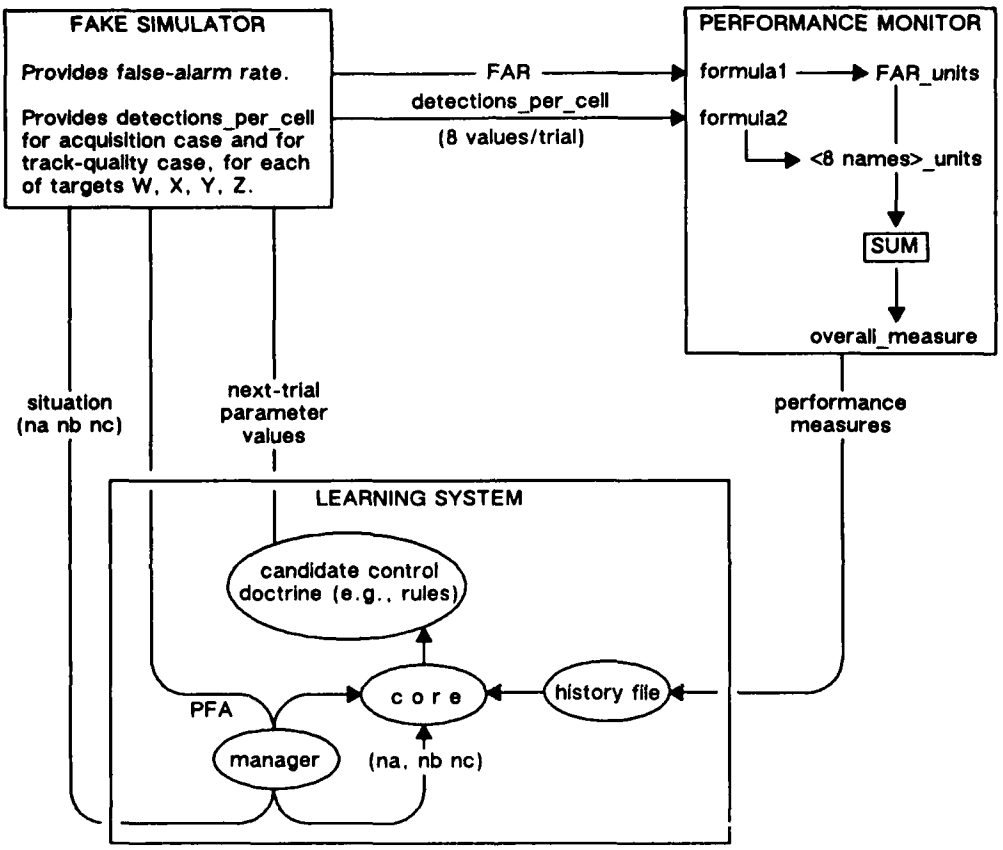

Figure 2. Interaction of modules.

Terminolgy used in Figure 2 is as follows:

1 Trial: PFA value, (na nb nc) values, parameter selection

Parameter selection for simple strategy: (a2 b1 b2 c1 c2) values

formula1: FAR_units = 120 * exp(2 * ((log FAR) – 2.1))

formula2: < >_units = 170 * exp(–3.52636 * detections_per_cell)
if detections_per_cell > 1, or

= 8.55 * exp(–0.5365 * detections_per_cell)
otherwise.

< > = targ_W_acquis, targ_X_acquis, targ_Y_acquis, targ_Z_acquis, track_W_qual, track_X_qual, track_Y_qual, track_Z_qual

EXAMPLE: CLUSTERING, TASK A

– for simple strategy: (na nb nc —> a1 a2 b1 b2 c1 c2)

Before beginning experiments, the following procedure was devised:

(1) Start with an "average" situation, e.g., na = 28, nb = 3, nc = 5.

(2) Select 4 parameter combinations for initial sampling, e.g., select some having the specified characteristics:

s1: All or most parameters maximized;

s2: All or most parameters minimized;

s3: Mixture of large and small; and

s4: Mixture of large and small, differing in at least two parameters.

(3) Form all possible pairs of samples (parameter combinations and performance): (s1 s2), (s1 s3), (s1 s4), (s2 s3), (s2 s4), (s3 s4).

(4) Divide these into 4 classes (permute pair if needed):

(unsat, unsat) — unsatisfactory if a measure exceeds its limit

(sat, unsat)

(sat, equiv sat)

(sat, better sat).

(5) Define a number of statements. For example:

STATEMENT TYPE	POSSIBLE INSTANCES
{parameter = value}	{a2 = .1} {a2 = .15} {a2 = .2} {b1 = 1} {b1 = 2} ... {c2 = .1} {c2 = .15} {c2 = .2}
{parameter product = value}	{a1*a2 = .1} .15} .2} {b1*b2 = .1} .15} .2} .3} .4} {c1*c2 = .1} .15} .2} .3} .4}
{quantized scan_rate = value}	{qsr = 1} 2} 3} 4}

(6) Let scan_rate = a2*na + b1*b2*nb + c1*c2*nc for the last statement type. The number of quantization levels is arbitrary.

Form 4 classes, and weight the statement instances as shown:

CLASS	ACTION
(unsat, unsat)	Assign –1 to every statement that occurs in both.
(sat, unsat)	Assign –1 to every statement that occurs only in unsat. Assign +1 to every statement that occurs only in sat.}
(sat, equiv sat)	Assign +1 to every statement that occurs in both.
(sat, better sat)	Assign +1 to every statement that occurs only in the better.

(7) Sum the scores. Statements with highest scores are best. Statements with lowest scores are poorest.

(8) Form a candidate rule set for selecting parameter combinations having good statements and no bad statements (or the closest to this situation).

(9) Use these to select the next sample. Perform new comparisons and weightings, refine the rules, and repeat the process (several times, if needed).

(10) To build rules for another situation, begin with the above rules modified by the initial knowledge:

Gradually decrease/increase a1*a2 as na increases/decreases,

decrease/increase b1*b2 as nb increases/decreases,

decrease/increase c1*c2 as nc increases/decreases.

(11) Refine these rules by taking several samples using these rules, then forming classes and weighting statements as above.

This procedure was tried for five sets of four pairs, using the statement type {parameter = value}. The formulas in Appendix A were used to calculate the performance measures, using a weighted sum for overall measure (the acquisition measures received weight 0.25 since they tend to be much greater than track quality measures). Track quality measures exceeding 100 and acquisition measures exceeding 125 were declared unsatisfactory. The results, using the scoring system above, were in the ballpark of the optimum combination (already calculated) but were not entirely

satisfactory. A major problem was that most of the unsatisfactory cases had good (low) overall measures, and the scoring system did not allow their values of overall measure to be a factor. A second scoring system was devised that differed in that two kinds of scores were summed — one based on overall measure and the other based on the (unsat, unsat) and (sat, unsat) pairs (i.e., an (unsat, sat) pair was additionally scored as a (sat, better unsat) when the unsat case had a lower overall measure. This gave better results, but none of the experiments resulted in the optimum combination.

Other statement types were investigated. The overall measure versus scan time was plotted for 16 parameter combinations, to see if scan time is a good indicator of the overall measure. (Scan time can be estimated without running a lengthy simulation, so would work well as an indicator if it correlates with performance. The result was a jagged curve that tended to be highest (poorest) for low scan times (corresponding to combinations with a2 = 0.1) and lowest (best) for medium scan times (corresponding to combinations with a2 = 0.15). This was encouraging, since the optimum combination includes a2 = 0.15, but the jagged behavior of the relationship could lead the learning system astray if an unfortunate set of initial parameter combinations is chosen. Figure A6 in Appendix A shows the graph of the overall measure versus the scan time for a typical situation choice (na, nb, nc) = (28, 3 ,5) and for various values of b1, b2, c1, and c2. Figure A5 illustrates performance measure values for parameters resulting in the minimum value of the overall measure, for the same typical situation.

Several variations of this clustering method were examined, but none produced entirely satisfactory results. With continued investigation we could probably find a set of statement types and scoring techniques that produce good results, but this would not be consistent with our desire to build a learning system that requires minimal initial knowledge.

FINDINGS AND ACCOMPLISHMENTS

SYSTEM STRUCTURE

After providing exact descriptions of the performance measures and completing the specification of an agile–beam radar problem, we completed the construction of a production system for setting parameters for this simplified model of a radar system. This time the system was built within the framework of a more complex learning production system using the flexibility of the PRISM production system architecture builder.

The representation to be learned was chosen to be a production system architecture because of its transparency to human users who may see the results as a doctrine described in rules. The learned material is easily broken up into comprehensible units (the individual rules) which can be separately analyzed and related to goals. The modularity of rules also permitted incremental development within a working (functioning) system.

PRISM was chosen as the architecture builder because it is capable of describing a very comprehensive space of interesting production system architectures. PRISM facilitated experimentation with architectures for learning, and eliminated some of the routine programming involved in implementing new production system languages.

The full production system for learning is a cluster of production memories, each with its own specialized task. Initially, a production memory called the INPUT–SITUATION–PM accepts user input and initializes the declarative memories (short-term memories) needed to process the description of the situation for the case that is input. The format for user input is specified exactly in Appendix E in Backus–Naur Form. No attempt was made to have user–friendly input or output formats at this stage since we developers were the only users.

SYSTEM GOALS

The user has the option of specifying different goals depending on whether the user wishes to merely find some solution to the parameter–setting problem (GOAL SATISFYING), to find the optimum solution (GOAL OPTIMIZING), the system to improve its ability to pick a near optimum setting on the first attempt (GOAL LEARNING), or to obtain a detailed monitoring of every decision made in the choice of parameters (GOAL MONITORING).

GOAL SATISFYING. When the goal is to satisfy the conditions on the problem, the most recently learned productions for setting parameters are applied to the problem to return a single solution along with ten measures of its performance. Two production memories are activated, SATISFYING–PM and PERFORMANCE–CHECK–PM.

GOAL OPTIMIZING. When the goal is to obtain an optimum solution, a series of such solutions are reported, with each subsequent one an improvement over the previous on its overall performance measure. The initial solution in the series is generated by activating the SATISFYING–PM and PERFORMANCE–CHECK–PM, but subsequent trials are determined by using the TRY–ANYTHING–PM along with those two long–term production memories. Productions for optimizing were originally housed in a separate memory, but efficiency considerations led us to incorporate them in with the productions for checking on performance. Although only parameter choices with improved performance are reported, all choices that are examined are saved to be used during learning.

GOAL MONITORING. When the goal is to monitor, the system assumes an optimum solution is desired and proceeds as in goal optimizing to produce a series of solutions. The only difference between goal monitoring and goal optimizing is in what the system reports. In goal monitoring, every decision and action along the path to obtaining each solution is reported by outputting each production rule that fires along with the situation in declarative memory that satisfied the conditions of the rule. Thus, every detail of the considerations leading to the choices of parameter sets is documented.

GOAL LEARNING. When the goal is to learn, parameter–setting cases are compared pairwise, spans of equivalent values (equivalent from the point of view of what parameters would be recommended) among the situation variables are determined, and new productions are generated that reflect observed relations among the situation variables when certain choices of parameters have better performance measures. The productions for learning are housed in the LEARN–PM, although much of the processing is accomplished in Interlisp since PRISM allows any arbitrary Lisp program to interface to the structures in it. The newly generated productions are added to the production memory for satisfying the constraints on a solution. Appendices F and G, respectively, contain the text of one of the learning programs and the log of a session on the Xerox 1108 of user interaction with the program.

RESULTS AND CONCLUSIONS

The conceptual clustering system originally constructed for experimentation for this project was not elaborated. The system uses a model–driven method for constructing nodes and minimizes the number of nodes in the clustering tree. However, a minimal number of nodes in the tree is neither a sign of ease of access nor a sign of there being a minimal number of tests for decisions made using the tree. Since additional research on conceptual clustering revealed far superior work in the area, minor improvements in our own conceptual clustering system were not attempted.

Some alternative approaches under consideration were found to be inferior. This included phrasing the performance system rules in terms of suggesting changes to the current parameter choices, and decomposing the performance system into separate subsystems for setting individual parameters.

Task B (learning to change parameter settings) was somewhat misguided, and we abandoned it for the following reasons:

(1) Nothing is lost by a 2–by–2 case comparison for setting parameters in terms of obtaining feedback to the learner.

(2) Unnecessary additional complexity for rules.

(3) Optimum (best) choices need not be near alterations of current choices. (Best guesses are always the rational choice.)

(4) Agile–beam radar situation facilitates radical changes. (Radical change may be an advantage for hostile elements attempting to second guess our trackings of them.)

From the generate and test view of learning, our key insight was to build into the generator as much of the test as possible. The learning subsystem constructs additional productions to incorporate more and more of the constraints on solutions or the known improvement in choices. Adding conditions in a given production, places limitations on when the possible candidates will be considered as suitable choices of parameters. This minimizes the search required to obtain good solutions. Each check of a candidate solution requires an elaborate calculation of performance measures, so transferring complexity to the productions is warranted. Eventually, the subsystem that returns the first solution to satisfy the constraints on a solution will do almost as well as the optimizing subsystem.

The development of the learning system was driven by experiment and examples. Boundaries on the applicability of the resulting system are a matter of human judgment. Research is still needed to construct the basis for a coherent theory of machine learning. A new field, computational learning theory, will contribute the analytical foundations. In addition to providing new learning models, extensions of existing models, and theoretical comparisons among learning models, the new field will contribute general learnability and nonlearnability results in existing models and upper and lower bounds on the required resources (time, space, number of examples). Analyses of specific learning algorithms will reveal their convergence–rates, resource requirements, noise sensitivity, stability, and robustness. We would like to direct future efforts to this field.

REFERENCES

Anderson, J.A., and G.E. Hinton. 1981. "Models of Information Processing in the Brain." Hinton, G.E. and Anderson, J.A. (Eds.), *Parallel Models of Associative Memory*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Anderson. J.R. 1983. *The Architecture of Cognition*. Harvard University Press, Cambridge, MA.

Anderson, J.R. 1986. "Knowledge Compilation: The General Learning Mechanism." Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. (Eds.), *Machine Learning: An Artificial Intelligence Approach*, vol. 2. Morgan–Kaufmann, Los Altos, CA.

Barto, A.G. 1985. *"Learning by Statistical Cooperation of Self-interested Neuron-Like Computing Elements."* Technical Report COINS 85–11 University of Massachusetts, Amherst.

Brown, J.S. 1973. "Steps Toward Automatic Theory Formation," *Proceedings of the Third International Joint Conference on Artificial Intelligence*, pp. 20–23. Stanford, CA.

Carbonell, J.G. 1986. "Derivational Anology: A Theory of Reconstructive Problem Solving and Expertise Acquisition." Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. (Eds.), *Machine Learning: An Artificial Intelligence Approach*, vol. 2. Morgan–Kaufmann, Los Altos, CA.

DeJong, G. and R. Mooney. 1986. "Explanation–Based Learning: An Alternative View," *Machine Learning*, vol. 1, no. 2, pp. 145–176.

Dietterich, T.G. and R.S. Michalski. 1983. "A Comparative Review of Selected Methods for Learning From Examples." Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. (Eds.), *Machine Learning: An Artificial Intelligence Approach*. Tioga Press, Palo Alto, CA.

Dillard, R.A. December, 1986. *Machine Self-teaching for Parameter Optimization*. NOSC Document 1039. Naval Ocean Systems Center, San Diego, CA.

Falkenhainer, B. 1985. "Proportionality Graphs, Units Analysis, and Domain Contraints: Improving the Power and Efficiency of the Scientific Discovery Process," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 552–554, University of California, Los Angeles.

Fisher, D. 1984. *A Hierarchical Conceptual Clustering Algorithm*. Technical Report, Department of Information and Computer Science, University of California, Irvine.

Fisher, D. and P. Langley. 1985. "Approaches to Conceptual Clustering," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 691–697, University of California, Los Angeles.

Fisher, D. 1986. *The Acquisition and Recognition of Basic Level Concepts*. Technical Report, Department of Information and Computer Science, University of California, Irvine.

Fisher, D. and P. Langley. 1986. "Methods of Conceptual Clustering and Their Relation to Numerical Taxonomy." Gale, W. (Ed.), *Artificial Intelligence and Statistics*.

Forgy, C.L. 1979. *On the Efficient Implementation of Production Systems.* Ph.D. Diss., Department of Computer Science, Carnegie–Mellon U., Pittsburgh, PA.

Fukushima, K. and S. Miyake. 1982. "Neocognitron: A New Algorithm for Pattern Recognition Tolerant of Deformations and Shifts of Position," *Pattern Recognition* vol. 15, no. 6, pp. 455–469.

Gluck, M. and J. Corter. 1985. "Information, Uncertainty, and the Utility of Categories," *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, pp. 283–287.

Hampson, S.E. and D.J. Volper (in press). "Linear Function Neurons: Structure and Training," *Biological Cybernetics*.

Holland, J.H. 1986. Escaping Brittleness: The Possibilities of General–Purpose Learning Algorithms Applied to Parallel Rule–Based Systems. Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. (Eds.), *Machine Learning: An Artificial Intelligence Approach*, vol. 2. Morgan–Kaufmann, Los Altos, CA

Iba, G. 1985. "Learning by Discovering Macros in Puzzle Solving," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 640–642.

Kohonen, T., E. Oja, and P. Lehtio. 1981. "Storage and Processing of Information in Distributed Associative Memory Systems." Hinton, G.E. and Anderson, J.A. (Eds.), *Parallel Models of Associative Memory*. Lawrence Erlbaum Associates, Hillsdale, NJ

Korf, R.E. 1982. "A Program That Learns to Solve Rubik's Cube," *Proceedings of the National Conference on Artificial Intelligence*, pp.164–167.

Kuchinski, M.J. 1985. *Battle Management Systems Control Rule Optimization Using Artificial Intelligence.* NSWC/MP–84–329 SBI–AD–F350 047, Naval Surface Weapons Center, Dahlgren, VA.

Laird, J.E., P.S. Rosenbloom, and A. Newell. 1986 "SOAR: The Anatomy of a General Learning Mechanism," *Machine Learning*, vol. 1, no. 1, pp. 11–46.

Langley, P., G.L. Bradshaw, and H.A. Simon. 1983. "Rediscovering Chemistry With the BACON System." Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. (Eds.), *Machine Learning: An Artificial Intelligence Approach.* Tioga Press, Palo Alto, CA

Langley, P. 1985. "Learning to Search: From Weak Methods to Domain–Specific Heuristics," *Cognitive Science*, vol. 9, pp. 217–260.

Langley, P., J. Zytkow, H.A. Simon, and G.L. Bradshaw. 1986. "The Search for Regularity: Four Aspects of Scientific Discovery." Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. (Eds.), *Machine Learning: An Artificial Intelligence Approach*, vol. 2. Morgan–Kaufmann, Los Altos, CA.

Lenat, D.B. 1977. "Automated Theory Formation in Mathematics," *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pp. 833–842, Cambridge, MA.

Michalski, R.S. and R. Stepp. 1983. "Learning from Observation: Conceptual Clustering." Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. (Eds.), *Machine Learning: An Artificial Intelligence Approach*. Tioga Press, Palo Alto, CA.

Miller, G.A. 1956. "The Magic Number Seven, Plus Or Minus Two: Some Limits On Our Capacity for Processing Information," *Psychological Review*, 63, pp. 81–97.

Minsky, M. 1967. *Computation: Finite and Infinite Machines*, Prentice Hall, Englewood Cliffs, NJ.

Mitchell, T.M. 1977. "Version Spaces: A Candidate Elimination Approach to Rule Learning," *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pp. 305–310, Cambridge, MA.

Mitchell, T.M. 1982. "Generalization as Search," *Artificial Intelli*gence, 18, pp. 203–226.

Mitchell, T.M., P. Utgoff, and R.B. Banerji. 1983. "Learning Problem Solving Heuristics by Experimentation." Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. (Eds.), *Machine Learning: An Artificial Intelligence Approach*. Tioga Press, Palo Alto, CA.

Mitchell, T.M., R.M. Keller, and S.T. Kedar–Cabelli. 1986. "Explanation–Based Generalization: A Unifying View," *Machine Learning 1*, pp 47–80.

Neves, D.M., and J.R. Anderson. 1981. "Knowledge Compilation: Mechanisms for the Automatization of Cognitive Skills." Anderson, J.R. (Ed.), *Cognitive Skills and Their Acquisition*, Lawrence Erlbaum Associates, Hillsdale, NJ.

Ohlsson, S. 1983. "A Constrained Mechanism for Procedure Learning," *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pp. 426–428, Karlsruhe, Germany.

Ohlsson, S. and P. Langley. 1986. *PRISM Tutorial and Manual.* Department of Information and Computer Science, University of California, Irvine.

Quinlan, J.R. 1983. "Learning Efficient Classification Procedures and their Application to Chess End Games." Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. (Eds.), *Machine Learning: An Artificial Intelligence Approach*. Tioga Press, Palo Alto, CA.

Rumelhart, D.E., G.E. Hinton, and R.J. Williams. 1985. *Learning Internal Representations by Error Propogation*. Technical Report 8506, Institute of Cognitive Science, University of California, San Diego.

Samuel, A.L. 1959. "Some studies in Machine Learning Using the Game of Checkers," *IBM Journal of Research and Development*, 3, pp. 210–229.

Silver, B. 1986. *Meta–Level Inference*. Elsevier Science Publishing Co., New York, NY.

Simon, H.A. and G. Lea. 1974. "Problem Solving and Induction: A Unified View." Gregg, L.W. (Ed.), *Knowledge and Cognition*. Lawrence Erlbaum, Potomac, MD.

Utgoff, P.E. 1984. *Shift of Bias for Inductive Concept Learning*. PhD. Diss., Rutgers University, Brunswick, NJ.

Vere, S. 1980. "Multilevel Counterfactuals for Generalizations of Relational Concepts and Productions," *Artificial Intelligence*, 14, pp.138–164.

Winston, P.H. 1975. "Learning Structural Descriptions from Examples." Winston, P.H. (Ed.), *The Psychology of Computer Vision*. McGraw–Hill, New York,

GLOSSARY

Analogical inference: Mapping information from a known entity (event, object, process, situation) to a similar one that is not so well known.

Attribute: A variable or one–argument descriptor used to characterize an object, event, entity, or process. Example: scan rate for a radar.

Causal analysis: Tracing the probable causes of observed events (occasionally used in credit assignment).

Characteristic description: A concept description that states properties characterizing all instances of a given concept or class.

Characterization: A description that identifies the attributes of members of a given class or instances of a concept.

Chunking: Grouping lower–level descriptions (patterns, operators, goals) into higher–level descriptions.

Composition: Assembling a sequence of production rules or operators into a single rule or operator.

Concept acquisition: Learning from examples.

Concept description: A symbolic data structure defining a concept describing all known instances of the concept.

Concept formation: A form of learning in which the learner generates concepts useful in characterizing a given collection of objects or facts or a subset of them.

Conceptual clustering: A form of learning from observation concerned with arranging entities (objects, events, situations, facts, processes, etc.) into classes characterized by simple descriptive concepts rather than into classes defined solely by a predefined measure of similarity among their members, as in conventional clustering.

Conditions: Left sides of production rules to be matched to representations of situations in working memory.

Constraint: A property or relation that restricts the space of possible solutions to a problem.

Constructive induction: An inductive learning process that generates new descriptors not provided in the description of initial facts or observations.

Credit (blame) assignment: Identifying the steps (decisions, operators, etc.) chiefly responsible for a success (failure) in the overall process of achieving a goal.

Derivational analogy: A case–based problem solving method in which derivations of solutions to similar problems are replayed and modified to solve new problems.

Description: A symbolic data structure defining a concept describing all known instances of the concept.

Descriptor: An attribute, function, (n–ary) relation, or predicate used as an elementary concept for describing entities (objects, events, situations, processes, etc.).

Discriminant description: A concept description that identifies properties that distinguish the given concept from other concepts under consideration.

Discrimination: Any method for moving from more general to more specific descriptions, that is, a method for systematically searching the space of descriptions in a general–to–specific manner (the opposite of "generalization," second meaning).

Domain of a descriptor: The set of admissible values that a descriptor may take as a component of a concept description.

Explanation–based generalization: A method that uses domain knowledge that explains a given example as constraints on the construction of descriptive concepts that characterize that example.

Extentionally identify: Specify (all of) the instances (of a concept or term).

Feature: A variable or one–argument descriptor (an attribute) used to characterize an object, event, entity, or process.

Generalization:

(1) Any method for generating some rule or concept description from a set of instances. (A relation between inputs and outputs of a learning system — all learning systems carry out generalization in this sense of the term, even if they move from general to more specific descriptions.)

(2) Any method for moving from more specific descriptions to more general descriptions, that is, a process for systematically searching the space of descriptions in a specific–to–general manner (usually in response to new positive instances that were not matched by an earlier description). (This is the sense in which I use the term, except where it appears as part of a term in current use such as explanation–based generalization.)

Heuristics: Imperfect but useful pieces of knowledge employed in reasoning and problem–solving tasks.

Heuristic search: A problem–solving method for finding a sequence of operators that transforms an initial state into a desired goal state. Search strategies used to generate, test, and prune operator sequences. Intensional definition: Characterization.

Intensionally identify: Characterize.

Intermediate measures (intermediate features of conditions or parameters): Composite (macro) characteristics constructed of elementary or primitive attributes that combine to improve the "grain–size" of reasoning in the domain.

Instantiation: Specifying values to fill in the slots in a schematic concept.

Knowledge compilation: Translating knowledge from a declarative form that cannot be used directly into an effective procedural form.

Machine learning: A subdomain of artificial intelligence concerned with developing computational models of learning, that is, systems that improve over time.

Macro–operator: An operator composed of a sequence of more primitive operators. Appropriate macro–operators simplify problem solving by allowing larger steps to be taken towards a solution and reducing search.

Means–ends analysis: A problem–solving method that searches at each step for operators that maximally reduce the differences between the current state and a known goal state.

Metalevel inference: A technique in which control information expressed declaratively (usually in the form of explicit rules) is separated from factual information in order to form a metalevel. Inference at the metalevel induces inference at the object–level. Search at the object–level is replaced by search at the metalevel. (See Silver, 1986.)

Neural network: A network of neuron–like elements that performs some simple logical function, typically a logic threshold function (LTF) or a sigmoidal function. Minsky (1967) showed that networks of LTFs possess the computational power of a Turing machine.

Operationalization: Knowledge compilation.

Parameter: An independent variable whose value contributes to determining the functioning of a system (in this case a sensor or communication system).

Precondition analysis: A method principally for learning problem–solving strategies from worked examples, combining metalevel inference with concepts from the field of planning.

Predicate: A statement that is either true or false.

Proceduralization: The conversion of declarative information into procedural form.

Production rule: A condition–action pair, stating that the action is to be performed if the condition is matched or satisfied.

Proportionality graphs: An undirected graph in which nodes represent variables and edges represent the existence of either a direct or inverse qualitatively proportionality relation between the two variables.

Schema: A symbolic structure that can be filled in by specific information (instantiated) to create an instance of the generic concept represented by the structure.

Specialization: Any method for moving from more general to more specific descriptions, that is, a method for systematically searching the space of descriptions in a general–to–specific manner. (The opposite of "generalization" in its second meaning)

Structural description: A symbolic representation of objects and concepts based on descriptions of their parts and the relationships among them.

Transformational analogy: A problem–solving method in which a solution to a similar problem is incrementally modified into a solution to the new problem.

Units analysis: A technique that imposes constraints on the construction of new terms from old to assure that the new terms can have some reasonable interpretation. Example: meters may be divided by seconds but not added to seconds. (If one variable has as its units the power of that of another, they may be added once the second is raised to that power.)

Version space: A set of candidate concept descriptions that are consistent with the training data, the knowledge, and the assumptions of the concept learner. This set defines a partially learned concept and can be represented by its maximally general and maximally specific members.

Weak methods: General methods for problem solving applicable in the absence of specific knowledge of the problem domain. Examples: means–ends analysis and heuristic search.

APPENDIX A

A Two-Dimensional Air Radar Example

A TWO-DIMENSIONAL AIR RADAR EXAMPLE

A simplified version of the example given in Dillard (1986) for a two-dimensional air-search radar is described here. The simplifications are the following: (1) the pulse duration is 10 μsec instead of a choice between 2 μsec and 100 μsec, and (2) only three per-beam-position target situations are considered rather than the six described there. Besides these changes, an additional value of dwell time (0.15 sec) is allowed. The pertinent specifications of this radar are:

- agile fan beam, 10-degree beamwidth
- 10 μsec pulse duration
- 162 nmi instrumented range
- 300 pulses per sec
- dwell time per beam = 0.10 sec (30 pulses), 0.15 sec (45 pulses), or 0.20 sec (60 pulses)
- looks (beams) per beam position per scan = 1 or 2

A scan is completed when each beam position has had at least one look. Since the antenna scans 360 degrees with a 10-degree beamwidth, the total number of beam positions is 36. The range resolution corresponding to a 10 μsec pulse is 0.8094 nmi, resulting in about 200 range-resolution cells per beam.

SITUATION PER BEAM POSITION

In each of the 36 beam positions, one of three possible situations can occur.

Situation a: DEFAULT. No track is likely to continue in that beam, and there is no indication from intelligence or electronic support measures (ESM) that a target is likely to enter from that direction.

Situation b: HIGH SPEED. A target that could be in that beam is travelling at high speed.

Situation c: OTHER. At least one target is believed to be in that beam (but not a high-speed target) and/or intelligence from other sensors indicate a target might be entering from that direction at any time.

SITUATION PER SCAN

Recall that a scan is a sequence of looks (beam pointings) such that each beam direction receives at least one look. On any given scan, let

na = number of beam positions believed to have situation a,

nb = number of beam positions believed to have situation b,

and

nc = number of beam positions believed to have situation c.

Since there are 36 beam positions, the sum of these numbers is 36, i.e., na + nb + nc = 36. Figure A1 shows the possible situations that can occur on any given scan.

Based on the values of na, nb, and nc, values of the variable parameters are selected for the next scan. Since there are 36 beam positions, the number of possible situations is 37*38*/2 = 703. Most of these are unlikely to occur, so can be disregarded.

no. of DEFAULT beam positions (na)	no. of HIGH-SPEED beam positions (nb)	no. of OTHER beam positions (nc)	
36	0	0	\
35	0	1	|
34	0	2	|
.	.	.	|
.	.	.	> 37 combinations
.	.	.	|
0	0	36	/
35	1	0	\
34	1	1	|
.	.	.	> 36 combinations
.	.	.	|
.	.	.	|
0	1	35	/
34	2	0	\
.	.	.	|
.	.	.	> 35 combinations
.	.	.	|
0	2	34	/
.	.	.	.
.	.	.	.
.	.	.	.
1	35	0	2 combinations
0	35	1	
0	36	0	1 combination

Figure A1. Situation space.

PARAMETER SELECTION STRATEGIES

During each scan, the two parameters that can be varied for each beam position are (1) the number of looks (beams) in that direction during that scan, and (2) the dwell–time per look. The simplest strategy would map the situation (na nb nc) into a selection of values of the parameters a1 a2 b1 b2 c1 c2, shown in the chart below.

	per–beam–position parameter	
beam situation	looks_per_scan	dwell_time (seconds)
a	a1 = 1	a2 = 0.1, 0.15, or 0.2
b	b1 = 1 or 2	b2 = 0.1, 0.15, or 0.2
c	c1 = 1 or 2	c2 = 0.1, 0.15, or 0.2

Figure A2 gives the reasonable combinations of parameter values that can be chosen under this strategy. The goal is to determine the current target situation (i.e., determine for each beam position if the situation is a, b, or c) and to use the best parameter combination for that situation (i.e., to use the best mapping of the form: na nb nc —> a1 a2 b1 b2 c1 c2).

A serious problem with this simple strategy is that weak (long–range or small) targets can have a very low probability of initial detection. A better strategy may be to alternate the value of dwell_time, (e.g., alternate it between 0.1 sec and 0.2 sec in those beam positions which would, otherwise, continuously have a 0.1 sec dwell time). There are various ways of specifying rules for varying dwell time. We arbitrarily chose the following as an example.

The simplest strategy (dwell_time not alternated) calls for values of a2, b1, b2, c1, and c2 to be selected for the next scan, based on the current situation (na, nb, nc). Out of 108 possible combinations, all but the 42 listed here are eliminated by bias constraints b2 > = a2, c2 > = a2, b1 > = c1.

DEFAULT		HIGH-SPEED		OTHER	
looks	dwell_time	looks	dwell_time	looks	dwell_time
a1	a2	b1	b2	c1	c2
1	.1	1	.1	1	.1
1	.1	1	.1	1	.15
1	.1	1	.1	1	.2
1	.1	1	.15	1	.1
1	.1	1	.15	1	.15
1	.1	1	.15	1	.2
1	.1	1	.2	1	.1
1	.1	1	.2	1	.15
1	.1	1	.2	1	.2
1	.1	2	.1	1	.1
1	.1	2	.1	1	.15
1	.1	2	.1	1	.2
1	.1	2	.15	1	.1
1	.1	2	.15	1	.15
1	.1	2	.15	1	.2
1	.1	2	.2	1	.1
1	.1	2	.2	1	.15
1	.1	2	.2	1	.2
1	.1	2	.1	2	.1
1	.1	2	.1	2	.15
1	.1	2	.1	2	.2
1	.1	2	.15	2	.1
1	.1	2	.15	2	.15
1	.1	2	.15	2	.2
1	.1	2	.2	2	.1
1	.1	2	.2	2	.15
1	.1	2	.2	2	.2
1	.15	1	.15	1	.15
1	.15	1	.15	1	.2
1	.15	1	.2	1	.15
1	.15	1	.2	1	.2
1	.15	2	.15	1	.15
1	.15	2	.15	1	.2
1	.15	2	.2	1	.15
1	.15	2	.2	1	.2
1	.15	2	.15	2	.15
1	.15	2	.15	2	.2
1	.15	2	.2	2	.15
1	.15	2	.2	2	.2
1	.2	1	.2	1	.2
1	.2	2	.2	1	.2
1	.2	2	.2	2	.2

Figure A2. Parameter space.

Dwell–time alternation rule: Every default (situation a) beam position which has had a 0.1–sec dwell time for the previous da–1 scans will have a 0.2–sec dwell time on the next scan. Every default beam position will have a 0.2–sec scan on at least 1 of every da scans. Similarly, every beam position with situation b (high speed) will have a 0.2–sec scan on at least 1 of every db scans and every beam position with situation c will have a 0.2–sec scan at least 1 of every dc scans.

The mapping for this rule is (na nb nc) —> (a1 da b1 db c1 dc). It may be that db and dc are unity under this strategy.

A major problem is determining the current target situation. Rules are needed for determining whether the situation in a beam position is a, b, or c. The accuracy of the determination depends partly on the current strategy and associated parameter values. An optimum strategy would also have adaptive rules for acquiring and tracking weak targets. These rules could also be refined with learning techniques. Acquisition of a target conventionally requires two or more detections. Reasonable rules can be specified arbitrarily, but learning procedures should be applicable also to improving acquisition rules.

TARGET TYPES

Performance measures of target acquisition and track quality will be calculated for four kinds of targets:

Target_W: high–speed, weak target

Target_X: high–speed, average–strength target

Target_Y: average–speed, weak target

Target_Z: average–speed, average–strength target

A "weak target" could be an average–size aircraft at maximum range or a small target at midrange. An "average–strength target" could be an average–size target at midrange or a smaller/closer or larger/farther target. Arbitrarily, let "high speed" be 1000 knots and "average speed" be 400 knots. All are assumed to be radial targets; i.e., they remain in a single–beam position, flying directly toward the radar.

FORMULAS

Let FAR denote the false–alarm rate (alarms per hour) and PFA denote the probability of a false alarm per resolution–cell decision. For a constant FAR, the PFA in beams having a 0.2–sec dwell should be twice as great as for a 0.1–sec dwell, and should be 1.5 times as great for a 0.15–sec dwell. Since there are 200 resolution cells per beam position, we have

$$\begin{aligned} FAR &= 200 * 3600 \text{ (sec/hr)} * PFA / 0.1 \text{ sec} \\ &= 7.2E6 * PFA \end{aligned}$$

where PFA is the value of PFA for a 0.1–sec dwell time.

For the simplest strategy (dwell time value not alternated), the time to complete one full scan (in seconds) is

scan_time = a2*na + b1*b2*nb + c1*c2*nc

Under the dwell-time alternation strategy, an approximation to the scan time is

scan_time = 0.1 * [na + na/da + b1 * (nb + nb/db) + c1 * (nc + nc/dc)]

This approximation assumes that the dwell time alternates randomly among the beams or is staggered, rather than, e.g., all default beams switching to the 0.2-sec dwell on the same scan.

For any strategy, the minimum value of the scan time is 0.1 sec * 36 beam positions = 3.6 sec, and the maximum value is 0.2 sec * 2 looks * 35 + 0.2 sec = 14.2 sec. (If all beam positions receive 2 looks, then the scan really becomes 2 scans, by definition of a scan.)

The time that a radial target remains in any resolution cell is time_in_cell = range_resolution / speed.

Since the range resolution is 0.8094 nmi, the value for a high-speed target is

time_in_cell = 0.8094 * 3600 / 1000 = 2.914 sec

and for an average_speed target is

time_in_cell = 0.8094 * 3600 / 400 = 7.2846 sec.

The average number of looks at a target while it remains in a single resolution cell is

looks_per_cell = looks_per_scan * time_in_cell / scan_time.

To compute the maximum value of looks_per_cell, we assume an average-speed target, a 0.1-sec dwell time in every beam position, and 2 looks in only that one beam position, giving scan_time = 0.1 sec * (35 + 2) = 3.7 sec. The maximum looks_per_cell is then 2 looks * (7.2846 sec/cell) / 3.7 sec = 3.94 looks. To compute the minimum value of looks_per_ cell, we assume a high-speed target that has not yet been detected, with 2 looks in all beam positions except this one and a 0.2-sec dwell_time in all positions. The scan_time is then 35 * (0.2 sec * 2 looks) + 0.2 sec = 14.2 sec, and the minimum looks_per_cell is (2.914 sec/bin) / (14.2 sec/scan) = 0.2052 looks. The value range of looks_per_cell is, therefore, roughly 0.2 to 4.0.

For the simple strategy, the average number of times a target will be detected while in a single resolution cell is

detections_per_cell = looks_per_cell * detection_probability.

The detection probability (per look) is a function of the false–alarm probability, the dwell time, and the target strength. For a fixed value of PFA, the detection probabilities need be computed (or read from a graph) for only six cases:

- 0.1–sec dwell, average strength
- 0.1–sec dwell, weak
- 0.15–sec dwell, average strength
- 0.15–sec dwell, weak
- 0.2–sec dwell, average strength
- 0.2–sec dwell, weak

For a false–alarm rate of 7.2 per hour (i.e., FAR = 7.2), the values of detection probability for these six cases, respectively, are 0.6456, 0.1365, 0.9315, 0.38417, 0.9921, and 0.6443, assuming noncoherent integration and a nonfluctuating target with a signal–to–noise ratio of 1 dB if average strength, and –1 dB if weak. (The probability of a false alarm is PFA = 0.000001 for a 0.1 sec dwell, PFA = 0.0000015 for a 0.15 sec dwell, and PFA = 0.000002 for a 0.2 sec dwell.)

Figure A3 lists formulas for detections_per_cell for each target type, for target acquisition, and track quality cases when using the simple strategy. Since the acquisition performance measures assume that the situation in the beam prior to detection of the target is situation a, the value of na used when computing track quality measures is greater by unity than the value for acquisition measures and the value of nb or nc is less by unity. (We assume that the situation in the other beams remains stable.)

For the dwell–time alternation strategy, the average number of detections per cell can be approximated by substituting for detection_probability the quantity: (Pd[0.2 sec dwell] + (d – 1) * Pd[0.1 sec dwell]) / d, where d is da, db, or dc for beam situations a, b, or c, respectively.

PERFORMANCE MEASURES

The component performance measures will be:

- FAR_units
- targ_W_acquis_units, targ_X_acquis_units, targ_Y_acquis _units, targ_Z_acquis_units
- track_W_qual_units, track_X_qual_units, track_Y_qual_units, track_Z_qual_units

The measure FAR_units is a function of the false–alarm rate. Acquisition measures assume that the values of the parameters used in that beam (looks_per_scan and dwell_time) are those for the default mode (a1 and a2 under the simple strategy). Track quality measures assume that the parameter values used in that beam are those for the high–speed mode if target_W or target_X, and those for the other mode if target_Y or target_Z. The target acquisition and track quality measures will be functions of the average number of detections of that target in a resolution cell (i.e., detections_per_cell).

DETECTIONS_PER_CELL

Target	Default Parameters (for acquisition cases)	Parameters Matched to Target (for track-quality cases)
W (fast)	$\frac{2.914 * Pd[a2, weak]}{scan\_time}$	$\frac{2.914 * b1 * Pd[b2, weak]}{scan\_time}$
X (fast)	$\frac{2.914 * Pd[a2, avg]}{scan\_time}$	$\frac{2.914 * b1 * Pd[b2, avg]}{scan\_time}$
Y	$\frac{7.285 * Pd[a2, weak]}{scan\_time}$	$\frac{7.285 * c1 * Pd[c2, weak]}{scan\_time}$
Z	$\frac{7.285 * Pd[a2, avg]}{scan\_time}$	$\frac{7.285 * c1 * Pd[c2, avg]}{scan\_time}$
$scan\_time = a2*na + b1*b2*nb + c1*c2*nc$		

Figure A3. Formulas for computing the values of detections_per_cell for each target type, for the target acquisition case, and the track quality case. (Simple strategy – dwell time not alternated.)

To minimize computations for initial experiments, a single formula can be used for all target-acquisition and track-quality measures. We have devised the following formula.

$$<>\_units = 170 * \exp(-3.52636 * detections\_per\_cell) \quad \text{if } detections\_per\_cell > 1$$

$$= 8.55 * \exp(-0.5365 * detections\_per\_cell) \text{ otherwise.}$$

Although the value of FAR_units is not important in our experiments (for reasons discussed below), a formula that can be used is the following.

$$FAR\_units = 120 * \exp(2 * ((\log FAR) - 2.1))$$

The overall performance measure (overall_measure) can be the sum of the component performance measures or a weighted sum. Since the acquisition measures tend to be much greater than the track quality measures, these have been given weight 0.25 in learning experiments.

COMPUTATIONS

A reasonable approach is to select a few values of FAR within a practical range and use the learning system to determine the best set of control rules for each one. (The same or very similar set of rules likely will result from all practical values of FAR, so the result for one value could be a starting set of rules for the next.) The

performance for several values of FAR can then be compared in a final stage. The operator often will prefer to set the false–alarm rate to a value considered appropriate for the moment.

Sample values of track–quality performance measures for the single–target case are given in figure A4, for the simple strategy. The false–alarm rate assumed for this case is 7.2 alarms per hour. The acquisition measures for acquiring the respective targets (while using default parameters in all 36 beam positions) have the same values as the last set of track quality measures (the 1 look, 0.1–sec dwell case). It is important to note, though, that the acquisition measures computed for a given na, nb, and nc pertain to a new target of the respective type (in a beam position having default parameters), and not to the acquisition of any targets currently being tracked.

TARGET:	W (na nb nc) = (35 1 0)	X	Y (na nb nc) = (35 0 1)	Z
TRACK QUALITY 2 looks 0.2–sec dwell (3.9–sec scan)	(0.963) 5.70	(1.483) 3.86	(2.407) 2.35	(3.706) 1.17
TRACK QUALITY 2 looks 0.1–sec dwell (3.7 sec scan)	(0.215) 79.65	(1.017) 4.95	(0.538) 25.54	(2.542) 2.19
TRACK QUALITY 1 looks 0.2–sec dwell (3.7–sec scan)	(0.507) 28.401	(0.781) 10.81	(1.269) 4.33	(1.953) 3.00
TRACK QUALITY 1 looks 0.2–sec dwell (3.6–sec scan)	(0.110) 115.14	(0.523) 26.92	(0.276) 64.18	(1.306) 4.24

Figure A4. Values of detections_per_cell (in parentheses) and track quality measure for the single target case, assuming the simple strategy, a false alarm rate of 7.2 per hour, and a 0.1–sec dwell in the default beams. (Recall that performance is measured in units of rejection, so a low value is good.)

The situation (na, nb, nc) = (28, 5, 3) was chosen for most experiments. The optimum set of parameters was found to be a2 = 0.15, b1 = 2, b2 = 0.2, c1 = 1, and c2 = 0.2. Figure A5 gives a printer listing of component performance measures and the overall measure.

Performance Measure Values for Parameters with Optimum Performance

(na nb nc) =	28	3		5	
(a2 b1 b2 c1 c2) =	0.15	2.00	0.20	1.00	0.20
Scan time = 6.400001					
W:det/cell (acq) = .2933578					
X:det/cell (acq) = .4517155					
Y:det/cell (acq) = .7306764					
Z:det/cell (acq) = .1.129289					
W:det/cell = .5867156					
X:det/cell = .9034309					
Y:det/cell = .7333945					
Z:det/cell = .1.129289					
FAR_units = 9.996					
W_acqu_units = 60.41957			W_qual_units = 21.47367		
X_acqu_units = 34.56658			X_qual_units = 7.028518		
Y_acqu_units = 12.92509			Y_qual_units = 12.80179		
Z_acqu_units = 4.664908			Z_qual_units = 4.664908		
Overall_measure = 84.10893					

Figure A5. Performance measure values for parameters resulting in the minimum value of the overall measure, for the typical situation (na, nb, nc) = (28, 3, 5).

(Acquisition units are weighted by a factor of .25 for the overall measure.)

In the experiments involving clustering methods for task A, we examined the relationship between the overall performance measure and the scan rate. This relationship is illustrated in Figure A6 and is described under the section "Experiments."

Overall Measure Versus Scan Time (for a typical situation)

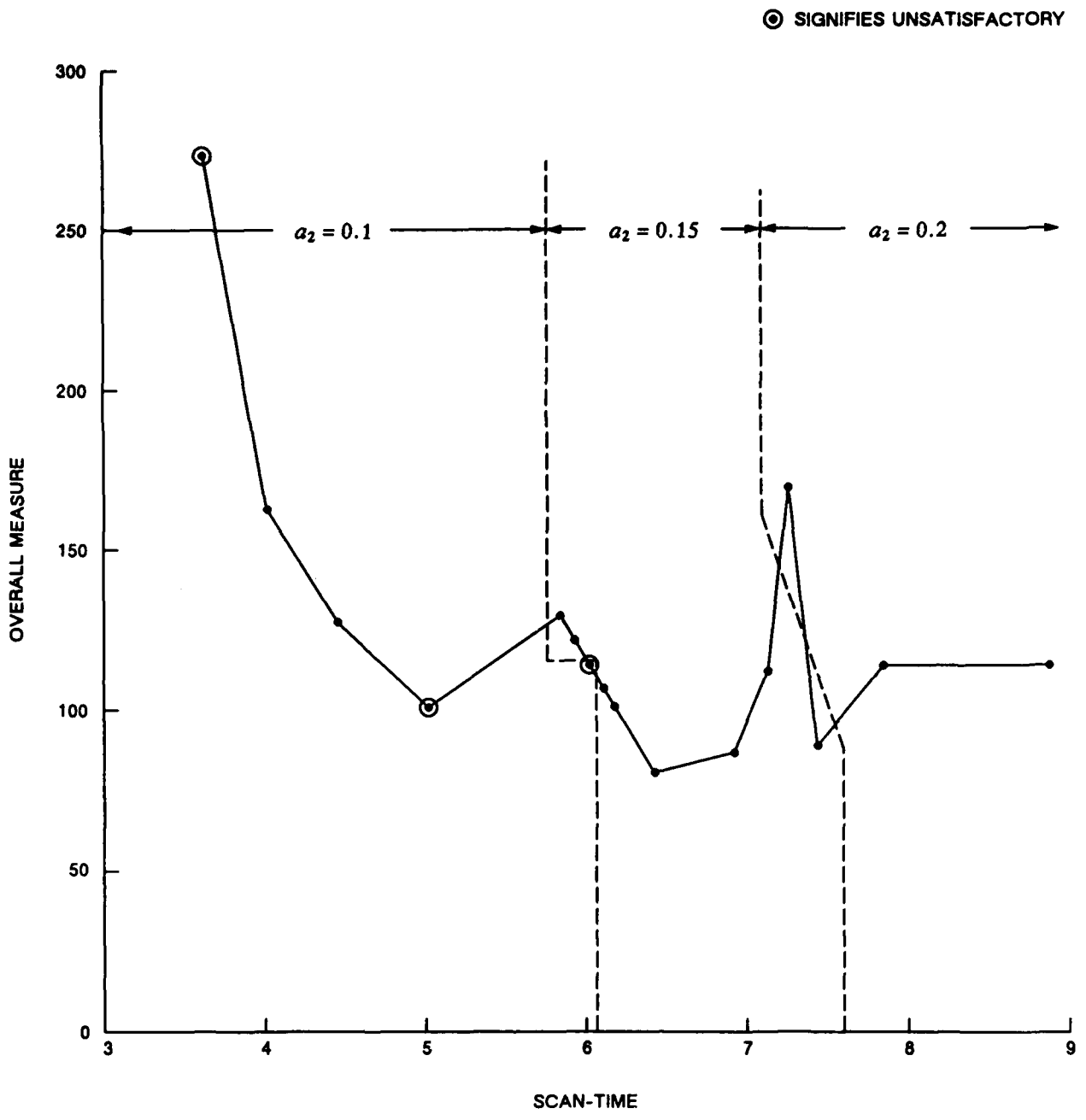

Figure A6. Overall measure versus scan time for (na, nb, nc) = (28, 3, 5) and for various values of a2, b1, b2, c1, and c2.

APPENDIX B

Preference Criteria for Hypotheses

PREFERENCE CRITERIA FOR HYPOTHESES

Simplicity of hypotheses

- of conditions
- of features and concepts used
- of actions or consequents.

Generality of hypotheses (more general hypotheses may be more economical as ways to compress information, but more specific ones tend to be closer to the data and do not go beyond it so recklessly).

Cost of using hypotheses for decisions

- cost of detecting features
- cost of establishing matches for the conditions (simplicity affects this)
  - number of tests
  - distribution of cases requiring multiple tests
    - (in a hierarchy of concepts which is organized to minimize the number of nodes, there could be a distribution of the cases which is less economical to process than if the concepts were organized in some other way)
- cost of conflict resolution between competing hypotheses.

Degree of approximation to the facts

- proportion of times the hypothesis applies correctly to the
  - number of times it applies
- vagueness of the hypothesis (in formats expressing numerical values
  - as mean and standard deviation, for example, a smaller standard deviation is preferred to a larger)
- precision of the prototype for concepts expressed as prototypes.

Interconnectivity to other hypotheses (using features which are common to a lot of other hypotheses may be preferable to using entirely new features since there will be a difference in what can be inferred).

Ease of determining what node an arbitrary case falls under.

APPENDIX C

Parameter Changes

PARAMETER CHANGES

How shall we describe changes in parameter settings? For the purpose of generating rules, we would like to have as few variants on the operators as possible. This poses a problem for the description of parameter changes because there are so many ways to make alterations in the parameter settings. The grainsize of the changes can be varied, not only within a metric (e.g., nominal, ordinal, interval, ratio, absolute), but descriptions of parameters can be revised and converted between metrics.

Operators for changing parameters could specify

I. Ordinal

large positive change

large negative change

no change

OR

large positive change

medium positive change

small positive change

no change

small negative change

medium negative change

large negative change

II. Metric (digital) units (These will have to be specific to a given parameter, based on understanding of the domain)

a. one unit at a time (greater, no change, or less)

b. n units at a time (greater, no change, or less)

c. logarithmic (or other power, 2**n, log(n), etc.) scale units at a time (greater, no change, or less).

The situation becomes all the more complex when multiple parameters are to be changed simultaneously. We are imposing constraints on when that will be considered to make the search tractable.

APPENDIX D

Terminating Conditions for the Learner

TERMINATING CONDITIONS FOR THE LEARNER

For a particular optimal parameter setting, the options are

a) Number of trials since the latest improvement,

b) Time of processing,

c) Exhausted the space of possibles to test, so we take the best found,

d) A preset performance goal within a certain epsilon distance,

e) A beam search of depth d revealing no improvements in performance, and

f) N seed conditions all having beam searches of depth d with no improvement.

For a complex multisensor system, a candidate goal recognizing rule could be:

IF you have radar r1,...,rn from which {ri} R have been selected, and for each ri selected, parameters pi1–pij are set, and for all performance measures each is below the maximum value allowable, and the performance measures meet the balance criterion

THEN you are finished.

When are we done learning parameter control doctrine? Probably never, but signs that we may be done would include:

Relative strengths on heuristic rules stabilize

No new rule variants are generated

All expected scenarios have been represented by instances (cases) presented to the learner.

APPENDIX E

Backus-Naur Form Specification of a Language for Inputting the Description of a Problem to the Learning Programs

BACKUS-NAUR FORM SPECIFICATION OF A LANGUAGE FOR INPUTTING THE DESCRIPTION OF A PROBLEM TO THE LEARNING PROGRAMS

<parameter-setting-task> ::= (<situation>) |
(<goal> <situation>)

<goal> ::= (GOAL <type>)

<type> ::= SATISFYING | {returns the first set of parameters which meet the constraints on a solution}

OPTIMIZING | {returns a series of sets of parameters, each with its associated performance measures. The next set in the series must be an improvement in performance over the previous on the overall performance measure}

LEARNING | {returns a series of sets of parameters as in the optimizing goal and subsequently improves both the optimizing and the satisfying production memories}

MONITORING {returns details of the considerations leading to the the choice of parameter sets}

<situation> ::= <na-nb-nc> |
<current-beam> |
<far> |
<far> <situation> |
<situation> <far> |
<current-beam> <situation> |
<situation> <current-beam> |
<na-nb-nc> <situation> |
<situation> <na-nb-nc>
<current-beam> ::= (CURRENT-BEAM <curr-beam>)

<curr-beam> ::= DEFAULT |
HIGH-SPEED |
<other>

<other> ::= OTHER

<far> ::= (FAR <decimal num>)

<na-nb-nc> ::= <num-for-na-nb-nc> |
<na-nb-nc> <num-for-na-nb-nc>

<num-for-na-nb-nc> ::= (na <num>) |
(nb <num>) |
(nc <num>)

<num> ::= 36 | 35 | 34 | 33 | 32 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 |
10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0

APPENDIX F

Text of the Learning Program, DISCRIM

TEXT OF THE LEARNING PROGRAM, DISCRIM

Listing of File {FLOPPY}DISCRIM , 3-Nov-87 12:17:18 by

```
(CREATE-COMPONENT CURRENT-CASE-WM
       INSTANCE-OF DECLARATIVE-MEMORY)

(CREATE-COMPONENT PREVIOUS-CASE-WM
       INSTANCE-OF DECLARATIVE-MEMORY)

(CREATE-COMPONENT CASE-WM
       INSTANCE-OF DECLARATIVE-MEMORY)

(CREATE-COMPONENT THIS-CASE-WM
       INSTANCE-OF DECLARATIVE-MEMORY)

(CREATE-COMPONENT BEST-PERFORMANCES-WM
       INSTANCE-OF DECLARATIVE-MEMORY)

(CREATE-COMPONENT TRIES-WM
       INSTANCE-OF DECLARATIVE-MEMORY)

(CREATE-COMPONENT LEARN-WM
       INSTANCE-OF DECLARATIVE-MEMORY)

(CREATE-COMPONENT CANDIDATE-RELATIONS-WM
       INSTANCE-OF DECLARATIVE-MEMORY)

(CREATE-COMPONENT OLD-CASE-WM
       INSTANCE-OF DECLARATIVE-MEMORY)

(CREATE-COMPONENT SATISFYING-PM
       INSTANCE-OF PROCEDURAL-MEMORY
       MATCHES-AGAINST TRIES-WM
       DEFAULT-ACTIONS (($ADD-TO WM ($RESULT))))

(CREATE-COMPONENT PERFORMANCE-CHECK-PM
       INSTANCE-OF PROCEDURAL-MEMORY
       MATCHES-AGAINST WM
       DEFAULT-ACTIONS (($ADD-TO WM ($RESULT))))

(CREATE-COMPONENT TRY-ANYTHING-PM
       INSTANCE-OF PROCEDURAL-MEMORY
       MATCHES-AGAINST TRIES-WM
       DEFAULT-ACTIONS (($ADD-TO TRIES-WM ($RESULT))))

(CREATE-COMPONENT INPUT-SITUATION-PM
       INSTANCE-OF PROCEDURAL-MEMORY
       MATCHES-AGAINST TRIES-WM
       DEFAULT-ACTIONS (($ADD-TO TRIES-WM ($RESULT))))

(CREATE-COMPONENT LEARNING-PM
       INSTANCE-OF PROCEDURAL-MEMORY
       MATCHES-AGAINST CURRENT-CASE-WM
       DEFAULT-ACTIONS (($ADD-TO LEARN-WM ($RESULT))))

(CREATE-COMPONENT LEARN-FROM-DIFFERENCES
       INSTANCE-OF DISCRIMINATION-PROCESS)
```

```
(CREATE-COMPONENT LEARN-FROM-SIMILARITIES
       INSTANCE-OF GENERALIZATION-PROCESS)


(BUILD-IN INPUT-SITUATION-PM

(START-PROBLEM
((START))
-->
(($CLEAR-DM WM TRIES-WM CURRENT-CASE-WM THIS-CASE-WM BEST-PERFORMANCES-WM)
 ($ADD-TO TRIES-WM ($GET-PROBLEM))
 ($ADD-TO TRIES-WM (FIRST-TRY) (NOT-ALTERNATIVES-SET))
 ($ADD-TO WM (NOT-ELIMINATED) (NOT-CASE-STORED) (NOT-PERF-KNOWN)) ))

(SET-MONITORING-TO-OPTIMIZE
((GOAL MONITORING))
-->
(($DELETE-FROM TRIES-WM (GOAL MONITORING))
 ($ADD-TO TRIES-WM (GOAL OPTIMIZING))
 ($DMTRACE FULL)
 ($PMTRACE FULL)))

(GO-TO-SATISFY
((FIRST-TRY))
-->
(($CALL SATISFYING-PM)
 ($CONTINUE)))

)


(BUILD-IN SATISFYING-PM

(SINGLE-HIGH-SPEED-TARGET
((NB 1) (NC 0) (@WM (B1 =B1) (B2 =B2) (C1 =C1)))
-->
(($DELETE-FROM WM (B1 =B1) (B2 =B2) (C1 =C1))
 ($ADD-TO WM (B1 2) (B2 0.2) (C1 1))))

(FEW-OTHER-TARGETS
((NC =NC) (NB 0) (*GREATER* =NC 0) (*GREATER* 4 =NC) (@WM (C1 =C1) (C2 =C2)))
-->
(($DELETE-FROM WM (C1 =C1) (C2 =C2))
 ($ADD-TO WM (C1 2) (C2 0.2))))

(HIGH-SPEED-TARGET
((CURRENT-BEAM HIGH-SPEED))
-->
(($ADD-TO WM (A2 0.15) (B1 2) (B2 0.2) (C1 2) (C2 0.15))))

(DEFAULT-SIT
((CURRENT-BEAM DEFAULT))
-->
(($ADD-TO WM (A2 0.15) (31 2) (B2 0.2) (C1 1) (C2 0.2))))

(OTHER-SIT
((CURRENT-BEAM OTHER))
-->
```

```
((SADD-TO WM (A2 0.15) (B1 2) (B2 0.2) (C1 2) (C2 0.2))))

(FINISHED-PICKING-FIRST-PARAM-SET
((@WM (A2 =A2) (B1 =B1) (B2 =B2) (C1 =C1) (C2 =C2)))
-->
((SADD-TO WM (FIRST-SET))
 (SADD-TO TRIES-WM (TRY (=A2 =B1 =B2 =C1 =C2)))))

(ELIMINATE-TRIAL-HIGH
((@WM (FIRST-SET) (A2 =A2) (B2 =B2) (*GREATER* =A2 =B2)
      (B1 =B1) (C1 =C1) (C2 =C2)))
-->
((SADD-TO WM (ELIMINATED))
 ($DELETE-FROM WM (NOT-ELIMINATED))
 ($DELETE-FROM TRIES-WM (TRY=A2 =B1 =B2 =C1 =C2)))
 ($ADD-TO TRIES-WM (TRIED=A2 =B1 =B2 =C1 =C2))
 ($DELETE-FROM WM (FIRST-SET))))

(ELIMINATE-TRIAL-OTHER
((@WM (FIRST-SET) (A2 =A2) (C2 =C2) (*GREATER* =A2 =C2)
      (B1 =B1) (B2 =B2) (C1 =C1)))
-->
((SADD-TO WM (ELIMINATED))
 ($DELETE-FROM WM (NOT-ELIMINATED))
 ($DELETE-FROM TRIES-WM (TRY=A2 =B1 =B2 =C1 =C2)))
 ($ADD-TO TRIES-WM (TRIED=A2 =B1 =B2 =C1 =C2))
 ($DELETE-FROM WM (FIRST-SET))))

(ELIMINATE-TRIAL
((@WM (FIRST-SET) (B1 =B1) (C1 =C1) (*GREATER* =C1 =B1)
      (A2 =A2) (B2 =B2) (C2 =C2)))
-->
((SADD-TO WM (ELIMINATED))
 ($DELETE-FROM WM (NOT-ELIMINATED))
 ($DELETE-FROM TRIES-WM (TRY=A2 =B1 =B2 =C1 =C2)))
 ($ADD-TO TRIES-WM (TRIED=A2 =B1 =B2 =C1 =C2))
 ($DELETE-FROM WM (FIRST-SET))))

(FIRST-SET-IN-SATISFYING
((GOAL SATISFYING) (@WM (FIRST-SET)))
-->
(($CALL PERFORMANCE-CHECK-PM)
 ($CONTINUE)))

(FOR-OPTIMIZING-OR-MONITORING
((@WM (FIRST-SET)) (GOAL OPTIMIZING))
-->
(($ADD-TO TRIES-WM ($IDENTIFY-ALTERNATIVES))
 ($DELETE-FROM TRIES-WM (NOT-ALTERNATIVES-SET))
 ($CALL PERFORMANCE-CHECK-PM)
 ($CONTINUE)))

(FOR-LEARNING
((@WM (FIRST-SET)) (GOAL LEARNING))
-->
(($ADD-TO TRIES-WM ($IDENTIFY-ALTERNATIVES))
 ($DELETE-FROM TRIES-WM (NOT-ALTERNATIVES-SET))
 ($CALL PERFORMANCE-CHECK-PM)
 ($CONTINUE)))
```

```
(BUILD-IN PERFORMANCE-CHECK-PM


(CHECK-PERFORMANCE
((A2 =A2) (NOT-PERF-KNOWN) (NOT-ELIMINATED)
 (B1 =B1) (B2 =B2) (C1 =C1) (C2 =C2)
 (@TRIES-WM (NA =NA) (TRY (=A2 =B1 =B2 =C1 =C2))
            (NB =NB) (NC =NC) (FAR =FAR)))
-->
(($ADD-TO WM ($CALCULATE-PERFORMANCE =NA =NB =NC =A2 =B1 =B2 =C1 =C2 =FAR))
 ($ADD-TO WM (PERF-KNOWN))
 ($DELETE-FROM WM (NOT-PERF-KNOWN)) ))

(STORE-CASE-FOR-LEARNING
((PERF-KNOWN) (NOT-CASE-STORED)
 (A2 =A2) (B1 =B1) (B2 =B2) (C1 =C1) (C2 =C2)
 (@TRIES-W(NA =NA) (NB =NB (NC =NC) (CURRENT-BEAM =B))
 (WACQ =WACQ) (XACQ =XACQ) (YACQ =YACQ) (ZACQ =ZACQ)
 (WQA =WQA) (XQA =XQA) (YQA =YQA) (ZQA =ZQA) (FARU =FARU) (OVAM =OVAM))
-->
(($ADD-TO CASE-WM ((=NA =NB =NC =B) (=A2 =B1 =B2 =C1 =C2)
                   (=WACQ =XACQ =YACQ =ZACQ =WQA =XQA =YQA =ZQA =FARU =OVAM)))
 ($ADD-TO THIS-CASE-WM ((=NA =NB =NC =B) (=A2 =B1 =B2 =C1 =C2)
       (=WACQ =XACQ =YACQ =ZACQ =WQA =XQA =YQA =ZQA =FARU =OVAM)))
 ($ADD-TO CURRENT-CASE-WM (CONDITIONS (=NA =NB =NC =B)))
   ADD-TO CURRENT-CASE-WM (SETTINGS (=A2 =B1 =B2 =C1 =C2)))
 ($ADD-TO CURRENT-CASE-WM (PERFORMANCES (=WACQ =XACQ =YACQ =ZACQ
                                         =WQA =XQA =YQA =ZQA =FARU =OVAM)))
 ($ADD-TO TRIES-WM (TRIED (=A2 =B1 =B2 =C1 =C2)))
 ($DELETE-FROM TRIES-WM (TRY (=A2 =B1 =B2 =C1 =C2)))
 ($ADD-TO WM (CASE-STORED))
 ($DELETE-FROM WM (NOT-CASE-STORED))))

(PERFORMANCE-UNACCEPTABLE
((CASE-STORED) (PERFORMANCE-BAD) (@TRIES-WM (GOAL SATISFYING)))
-->
(($ADD-TO WM (ELIMINATED))
 ($DELETE-FROM WM (PERFORMANCE-BAD))
 ($DELETE-FROM WM (NOT-ELIMINATED))))

(DECIDE-WHAT-CAN-BE-TRIED
((ELIMINATED) (@TRIES-WM (FIRST-TRY) (GOAL SATISFYING) (NOT-ALTERNATIVES-SET)))
-->
(($DELETE-FROM TRIES-WM (FIRST-TRY))
 ($ADD-TO TRIES-WM (NOT-FIRST-TRY))
 ($ADD-TO TRIES-WM ($IDENTIFY-ALTERNATIVES))
 ($DELETE-FROM TRIES-WM (NOT-ALTERNATIVES-SET))
))

(REMOVE-SET-TRIED-FIRST
((@TRIES-WM (GOAL SATISFYING) (TRIED =PARAMS) (TRY =PARAMS)))
-->
(($DELETE-FROM TRIES-WM (TRY =PARAMS))
 ))
```

```
(GET-NEW-TRIAL
((ELIMINATED)
 (@TRIES-WM  (TRY =PARAMS) (NOT-FIRST-TRY)) )
-->
(($CLEAR-DM WM)
 ($ADD-TO WM (NOT-ELIMINATED) (NOT-CASE-STORED) (NOT-PERF-KNOWN))
 ($ADD-TO WM ($GET-NEXT-TRY =PARAMS))
 ($DELETE-FROM TRIES-WM (TRY =PARAMS))))

(FINISHED-SATISFYING
((CASE-STORED) (PERFORMANCE-OK) (@TRIES-WM (GOAL SATISFYING)))
-->
(($WRITE "*****THE PARAMETERS FOR:  ")
 ($PRINT-FIRSTS TRIES-WM CURRENT-BEAM NA NB NC)
 ($WRITE "**ARE:  ")
 ($PRINT-FIRSTS WM A2 B1 B2 C1 C2)
 ($WRITE "**WITH PERFORMANCE VALUES:  ")
 ($PRINT-FIRSTS WM FARU WACQ XACQ YACQ ZACQ WQA XQA YQA ZQA OVAM)
 ($CLEAR-DM WM TRIES-WM)
 ($ADD-TO TRIES-WM (START))
 ($CALL INPUT-SITUATION-PM)
 ($CONTINUE)))

(FIRST-OPTIMIZING-TRY
((CASE-STORED) (@TRIES-WM (GOAL OPTIMIZING))
 (@TRIES-WM (FIRST-TRY)) (OVAM =OVAM))
-->
(($DELETE-FROM TRIES-WM (FIRST-TRY))
 ($ADD-TO TRIES-WM (NOT-FIRST-TRY))
 ($CLEAR-DM BEST-PERFORMANCES-WM)
 ($ADD-TO BEST-PERFORMANCES-WM (OVAM =OVAM))
 ($WRITECR "*****THE PARAMETERS FOR:  ")
 ($PRINT-FIRSTS TRIES-WM CURRENT-BEAM NA NB NC)
 ($WRITECR "**ARE:  ")
 ($PRINT-FIRSTS WM A2 B1 B2 C1 C2)
 ($WRITECR "**WITH PERFORMANCE VALUES:  ")
 ($PRINT-FIRSTS WM WACQ XACQ YACQ ZACQ WQA XQA YQA ZQA FARU OVAM)
 ($CALL TRY-ANYTHING-PM)
 ($CONTINUE)
))


(CHECK-FOR-OVERALL-BEST-SO-FAR
((CASE-STORED) (@TRIES-WM (GOAL OPTIMIZING))
 (@BEST-PERFORMANCES-WM (OVAM =BOVAM))
 (OVAM =OVAM)
 (#GREATER# =BOVAM =OVAM))
-->
(($CLEAR-DM BEST-PERFORMANCES-WM)
 ($ADD-TO BEST-PERFORMANCES-WM (OVAM =OVAM))
 ($WRITECR "*****THE PARAMETERS FOR:  ")
 ($PRINT-FIRSTS TRIES-WM CURRENT-BEAM NA NB NC)
 ($WRITECR "**ARE:  ")
 ($PRINT-FIRSTS WM A2 B1 B2 C1 C2)
 ($WRITECR "**WITH PERFORMANCE VALUES:  ")
 ($PRINT-FIRSTS WM WACQ XACQ YACQ ZACQ WQA XQA YQA ZQA FARU OVAM)
 ($CALL TRY-ANYTHING-PM)
 ($CONTINUE)
))
```

```
(BUILD-IN TRY-ANYTHING-PM

(TRY-ANYTHING-NOT-TRIED
((TRY =PARAMS))
-->
(($CLEAR-DM WM)
 ($ADD-TO WM (NOT-ELIMINATED) (NOT-CASE-STORED) (NOT-PERF-KNOWN)
            ($GET-NEXT-TRY =PARAMS))
 ($CALL PERFORMANCE-CHECK-PM)
 ($CONTINUE)))

)

(BUILD-IN LEARNING-PM

(COUNT-POINTS
((@CASE-WM (=PERFS1) (=PERFS2) (*NOT (*EQUAL =PERFS1 =PERFS2))))
-->
(($ADD-TO LEARN-WM ($COUNT-PNTS (=PERFS1 =PERFS2)))))

(COMPARE-CASES
((RELATION =REL BEST =PARAMETERS))
-->
(($BUILD-IN SATISFYING-PM
   (=NEW-PROD-NAME ((=REL)) --> ((/$ADD-TO WM (=PARAMETERS)))))
 ($DELETE-FROM LEARN-WM $1)))

)

(ADD-TO TRIES-WM (START))
(CALL INPUT-SITUATION-PM)

(DEXPR GET-PROBLEM ()  (PROG (PROBLEM PROB RESULT GOAL CURRENT-BEAM NA NB NC FAR)
       (PRINK "TYPE A PROBLEM (OR Q TO QUIT): ")
       (SETQ PROB (CAR (INPUT-LINE)))
       (COND ((EQUAL PROB 'Q) (HALT) (RETURN NIL)))
       (SETQ GOAL 'SATISFYING)
       (SETQ CURRENT-BEAM 'DEFAULT)
       (SETQ NA 36)
       (SETQ NB 0)
       (SETQ NC 0)
       (SETQ FAR 7.2)

 LOOP  (SETQ PROBLEM PROB)
       (COND ((EQ (CAAR PROBLEM) 'GOAL) (SETQ GOAL (CADAR PROBLEM)))
             ((EQ (CAAR PROBLEM) 'NA) (SETQ NA (CADAR PROBLEM)))
             ((EQ (CAAR PROBLEM) 'NB) (SETQ NB (CADAR PROBLEM)))
             ((EQ (CAAR PROBLEM) 'NC) (SETQ NC (CADAR PROBLEM)))
             ((EQ (CAAR PROBLEM) 'CURRENT-BEAM) (SETQ CURRENT-BEAM (CADAR PROBLEM)))
             ((EQ (CAAR PROBLEM) 'FAR) (SETQ FAR (CADAR PROBLEM)))
             (T ((WRITECR "*****ENTER PROBLEM IN SPECIFIED FORMAT*****")
                 (PRINK "TYPE A PROBLEM (OR Q TO QUIT): ")
                 (SETQ PROB (CAR (INPUT-LINE)))
                 (COND ((EQUAL PROB 'Q) (HALT) (RETURN NIL)))
                 (SETQ GOAL 'SATISFYING)
```

F-7

```
        (SETQ CURRENT-BEAM 'DEFAULT)
        (SETQ NA 36)
        (SETQ NB 0)
        (SETQ NC 0)
        (SETQ FAR 7.2) )) )
(SETQ PROB (CDR PROBLEM))
(COND ((NOT (NULL PROB)) (GO LOOP)) )

(COND ((NOT (EQP (PLUS NA NB NC) 36))
          ((WRITECR "****THE SUM OF NA, NB, AND NC MUST BE 36.****")
           (PRINK "TYPE A PROBLEM (OR Q TO QUIT): ")
           (SETQ PROB (CAR (INPUT-LINE)))
           (COND ((EQUAL PROB 'Q) (HALT) (RETURN NIL)))
           (SETQ GOAL 'SATISFYING)
           (SETQ CURRENT-BEAM 'DEFAULT)
           (SETQ NA 36)
           (SETQ NB 0)
           (SETQ NC 0)
           (SETQ FAR 7.2)
     (GO LOOP) ))

      ((NOT (OR (EQ GOAL 'SATISFYING)
                (EQ GOAL 'OPTIMIZING)
                (EQ GOAL 'LEARNING)
                (EQ GOAL 'MONITORING)))
       ((WRITECR "***GOAL MUST BE SATISFYING, OPTIMIZING, LEARNING, OR MONITORING.***")
       (PRINK "TYPE A PROBLEM (OR Q TO QUIT): ")
       (SETQ PROB (CAR (INPUT-LINE)))
       (COND ((EQUAL PROB 'Q) (HALT) (RETURN NIL)))
       (SETQ GOAL 'SATISFYING)
       (SETQ CURRENT-BEAM 'DEFAULT)
       (SETQ NA 36)
       (SETQ NB 0)
       (SETQ NC 0)
       (SETQ FAR 7.2)
     (GO LOOP) ))

      ((NOT (OR (EQ CURRENT-BEAM 'HIGH-SPEED)
                (EQ CURRENT-BEAM 'DEFAULT)
                (EQ CURRENT-BEAM 'OTHER)))
       ((WRITECR "***CURRENT-BEAM MUST BE HIGH-SPEED, DEFAULT OR OTHER.***")
           (PRINK "TYPE A PROBLEM (OR Q TO QUIT): ")
           (SETQ PROB (CAR (INPUT-LINE)))
           (COND ((EQUAL PROB 'Q) (HALT) (RETURN NIL)))
           (SETQ GOAL 'SATISFYING)
           (SETQ CURRENT-BEAM 'DEFAULT)
           (SETQ NA 36)
           (SETQ NB 0)
           (SETQ NC 0)
           (SETQ FAR 7.2)
     (GO LOOP) ))   )


 (PRISM:PUSH (LIST 'GOAL GOAL) RESULT)
 (PRISM:PUSH (LIST 'NA NA) RESULT)
 (PRISM:PUSH (LIST 'NB NB) RESULT)
 (PRISM:PUSH (LIST 'NC NC) RESULT)
 (PRISM:PUSH (LIST 'CURRENT-BEAM CURRENT-BEAM) RESULT)
 (PRISM:PUSH (LIST 'FAR FAR) RESULT)
```

```
        (RETURN RESULT)))


(DEXPR GREATER? (X Y)
  (AND (NUMBERP X) (NUMBERP Y) (GREATERP X Y)))

(DEXPR IDENTIFY-ALTERNATIVES ()
       (PROG (RESULT ALTERNS ALT)
       (SETQQ ALTERNS ((0.1 1 0.1 1 0.1) (0.1 1 0.1 1 0.15) (0.1 1 0.1 1 0.2)
                       (0.1 1 0.15 1 0.1) (0.1 1 0.15 1 0.15) (0.1 1 0.15 1 0.2)
                       (0.1 1 0.2 1 0.1) (0.1 1 0.2 1 0.15) (0.1 1 0.2 1 0.2)
                       (0.1 2 0.1 1 0.1) (0.1 2 0.1 1 0.15) (0.1 2 0.1 1 0.2)
                       (0.1 2 0.15 1 0.1) (0.1 2 0.15 1 0.15) (0.1 2 0.15 1 0.2)
                       (0.1 2 0.2 1 0.1) (0.1 2 0.2 1 0.15) (0.1 2 0.2 1 0.2)
                       (0.1 2 0.1 2 0.1) (0.1 2 0.1 2 0.15) (0.1 2 0.1 2 0.2)
                       (0.1 2 0.15 2 0.1) (0.1 2 0.15 2 0.15) (0.1 2 0.15 2 0.2)
                       (0.1 2 0.2 2 0.1) (0.1 2 0.2 2 0.15) (0.1 2 0.2 2 0.2)
                       (0.15 1 0.15 1 0.15) (0.15 1 0.15 1 0.2)
                       (0.15 1 0.2 1 0.15) (0.15 1 0.2 1 0.2)
                       (0.15 2 0.15 1 0.15) (0.15 2 0.15 1 0.2)
                       (0.15 2 0.2 1 0.15) (0.15 2 0.2 1 0.2)
                       (0.15 2 0.15 2 0.15) (0.15 2 0.15 2 0.2)
                       (0.15 2 0.2 2 0.15) (0.15 2 0.2 2 0.2)
                       (0.2 1 0.2 1 0.2) (0.2 2 0.2 1 0.2) (0.2 2 0.2 2 0.2)) )
        (SETQ ALT ALTERNS)

  LOOP (PRISM:PUSH (LIST 'TRY (CAR ALT)) RESULT)
       (COND ((NOT (NULL (CDR ALT))) (SETQ ALT (CDR ALT)))
  (GO LOOP)))
       (PRISM:PUSH '(ALTERNATIVES-SET) RESULT)
       (RETURN RESULT)))

(DEXPR COUNT-PNTS (X1 X2)
       (PROG (RESULT SIT SETS PERFS PERFSN
            XSI XSETS XPERFS
            NA NB NC BEAM
            XNA XNB XNC XBEAM
            A2 B1 B2 C1 C2       FARU
            XA2 XB1 XB2 XC1 XC2   XFARU
            WACQ XACQ YACQ ZACQ WAQ XAQ YAQ ZAQ OVAM
            XWACQ XXACQ XYACQ XZACQ
            XWAQ XXAQ XYAQ XZAQ XOVAM
            MAX-WACQ MAX-XACQ MAX-YACQ MAX-ZACQ MAX-WQA MAX-XQA MAX-YQA MAX-ZQA
            RELNA RELNB RELNC)

       (SETQ SIT (CAR X1))
       (SETQ SETS (CADR X1))
       (SETQ PERFS (CADDR X1))

       (SETQ XSIT (CAR X2))
       (SETQ XSETS (CADR X2))
       (SETQ XPERFS (CADDR X2))

    (COND ((EQUAL PERFS XPERFS)
            (COND ((GREATERP NA XNA)
                    (PRISM:PUSH (LIST 'SPAN (LIST  'NA XNA NA)) RESULT))
                  ((GREATERP XNA NA)
                    (PRISM:PUSH (LIST 'SPAN (LIST  'NA NA XNA)) RESULT)) )
            (COND ((GREATERP NB XNB)
```

```
                (PRISM:PUSH (LIST 'SPAN (LIST  'NB XNB NB)) RESULT))
              ((GREATERP XNB NB)
                (PRISM:PUSH (LIST 'SPAN (LIST  'NB NB XNB)) RESULT)) )
        (COND ((GREATERP NC XNC)
                (PRISM:PUSH (LIST 'SPAN (LIST  'NC XNC NC)) RESULT))
              ((GREATERP XNC NC)
                (PRISM:PUSH (LIST 'SPAN (LIST  'NC NC XNC)) RESULT)) )
        (RETURN RESULT) ))

    (SETQ NA (CAR SIT))
    (SETQ NB (CADR SIT))
    (SETQ NC (CADDR SIT))

    (SETQ A2 (CAR SETS))
    (SETQ B1 (CADR SETS))
    (SETQ B2 (CADDR SETS))
    (SETQ C1 (CADDDR SETS))
    (SETQ C2 (CADDDDR SETS))

    (SETQ OVAM (FLAST PERFS))

    (SETQ XNA (CAR XSIT))
    (SETQ XNB (CADR XSIT))
    (SETQ XNC (CADDR XSIT))

    (SETQ XA2 (CAR XSETS))
    (SETQ XB1 (CADR XSETS))
    (SETQ XB2 (CADDR XSETS))
    (SETQ XC1 (CADDDR XSETS))
    (SETQ XC2 (CADDDDR XSETS))

    (SETQ XOVAM (FLAST XPERFS))

    (SETQ BEAM (CADDDR SIT))
    (SETQ XBEAM (CADDDR XSIT))

    (SETQ WACQ (CAR PERFS))
    (SETQ XACQ (CADR PERFS))
    (SETQ YACQ (CADDR PERFS))
    (SETQ ZACQ (CADDDR PERFS))

    (SETQ PERFSN (CDDDDR PERFS))
    (SETQ WAQ (CAR PERFSN))
    (SETQ XAQ (CADR PERFSN))
    (SETQ YAQ (CADDR PERFSN))
    (SETQ ZAQ (CADDDR PERFSN))

    (SETQ MAX-WACQ 125) (SETQ MAX-XACQ 125) (SETQ MAX-YACQ 125) (SETQ
MAX-ZACQ 125)
    (SETQ MAX-WQA 100)  (SETQ MAX-XQA 100)  (SETQ MAX-YQA 100)  (SETQ MAX-ZQA 100)

    (SETQ OK (AND (LESSP WACQ MAX-WACQ)
                  (LESSP XACQ MAX-XACQ)
                  (LESSP YACQ MAX-YACQ)
                  (LESSP ZACQ MAX-ZACQ)
                  (LESSP WQA MAX-WQA)
                  (LESSP XQA MAX-XQA)
                  (LESSP YQA MAX-YQA)
                  (LESSP ZQA MAX-ZQA)))
```

```
      (SETQ PERFSN (CDDDDR PERFSN))
      (SETQ FARU (CAR PERFSN))

      (SETQ XWACQ (CAR XPERFS))
      (SETQ XXACQ (CADR XPERFS))
      (SETQ XYACQ (CADDR XPERFS))
      (SETQ XZACQ (CADDDR XPERFS))

      (SETQ PERFSN (CDDDDR XPERFS))
      (SETQ XWAQ (CAR PERFSN))
      (SETQ XXAQ (CADR PERFSN))
      (SETQ XYAQ (CADDR PERFSN))
      (SETQ XZAQ (CADDDR PERFSN))

     (SETQ XOK (AND (LESSP XWACQ MAX-WACQ)
                    (LESSP XXACQ MAX-XACQ)
                    (LESSP XYACQ MAX-YACQ)
                    (LESSP XZACQ MAX-ZACQ)
                    (LESSP XWQA MAX-WQA)
                    (LESSP XXQA MAX-XQA)
                    (LESSP XYQA MAX-YQA)
                    (LESSP XZQA MAX-ZQA)))

      (SETQ PERFSN (CDDDDR PERFSN))
      (SETQ XFARU (CAR PERFSN))


  (COND ((NOT (EQUAL SIT XSIT))
         (COND ((GREATERP NA XNA)
                (COND ((GREATERP XOVAM OVAM)
                       (PRISM:PUSH (LIST (LIST 'RELATION (LIST '(NA =NA) (LIST '#GREATERP '=NA XNA)) ) (LIST 'BEST (LIST 'A2 A2)
(LIST 'B1 B1) (LIST 'B2 B2) (LIST 'C1 C1) (LIST 'C2 C2))) RESULT))
                      ((GREATERP OVAM XOVAM)
                       (PRISM:PUSH (LIST (LIST 'RELATION (LIST '(NA =NA) (LIST '#LESSP '=NA (PLUS XNA 1))) ) (LIST 'BEST (LIST 'A
2 XA2) (LIST 'B1 XB1) (LIST 'B2 XB2) (LIST 'C1 XC1) (LIST 'C2 XC2))) RESULT))
                      (T (PRISM:PUSH (LIST 'SPAN (LIST  'NA XNA NA)) RESULT))))

               ((GREATERP XNA NA)
                (COND ((GREATERP OVAM XOVAM)
                       (PRISM:PUSH (LIST (LIST 'RELATION (LIST '(NA =NA) (LIST '#GREATERP '=NA NA)) ) (LIST 'BEST (LIST 'A2 XA2)
(LIST 'B1 XB1) (LIST 'B2 XB2) (LIST 'C1 XC1) (LIST 'C2 XC2))) RESULT))
                      ((GREATERP XOVAM OVAM)
                       (PRISM:PUSH (LIST (LIST 'RELATION (LIST '(NA =NA) (LIST '#LESSP '=NA (PLUS NA 1))) ) (LIST 'BEST (LIST 'A2
 A2) (LIST 'B1 B1) (LIST 'B2 B2) (LIST 'C1 C1) (LIST 'C2 C2)) ) RESULT))
                      (T (PRISM:PUSH (LIST 'SPAN (LIST  'NA NA XNA)) RESULT))))
         )

         (COND ((GREATERP NB XNB)
                (COND ((GREATERP XOVAM OVAM)
                       (PRISM:PUSH (LIST (LIST 'RELATION (LIST '(NB =NB) (LIST '#GREATERP '=NB XNB)) ) (LIST 'BEST (LIST 'A2 A2)
(LIST 'B1 B1) (LIST 'B2 B2) (LIST 'C1 C1) (LIST 'C2 C2))) RESULT))
                      ((GREATERP OVAM XOVAM)
                       (PRISM:PUSH (LIST (LIST 'RELATION (LIST '(NB =NB) (LIST '#LESSP '=NB (PLUS XNB 1))) ) (LIST 'BEST (LIST 'A
2 XA2) (LIST 'B1 XB1) (LIST 'B2 XB2) (LIST 'C1 XC1) (LIST 'C2 XC2))) RESULT))
                      (T (PRISM:PUSH (LIST 'SPAN (LIST  'NB XNB NB)) RESULT))))

               ((GREATERP XNB NB)
                (COND ((GREATERP OVAM XOVAM)
```

```
                        (PRISM:PUSH (LIST (LIST 'RELATION (LIST '(NB =NB) (LIST '#GREATERP '=NB NB)) ) (LIST 'BEST (LIST 'A2 XA2)
(LIST 'B1 XB1) (LIST 'B2 XB2) (LIST 'C1 XC1) (LIST 'C2 XC2)) ) RESULT))
                     ((GREATERP XOVAM OVAM)
                        (PRISM:PUSH (LIST (LIST 'RELATION (LIST '(NB =NB) (LIST '#LESSP '=NB (PLUS NB 1))) ) (LIST 'BEST (LIST 'A2
 A2) (LIST 'B1 B1) (LIST 'B2 B2) (LIST 'C1 C1) (LIST 'C2 C2)) ) RESULT))
                     (T (PRISM:PUSH (LIST 'SPAN (LIST  'NB NB XNB)) RESULT))))
          )

         (COND ((GREATERP NC XNC)
                (COND ((GREATERP XOVAM OVAM)
                        (PRISM:PUSH (LIST (LIST 'RELATIO  (LIST '(NC =NC) (LIST '#GREATERP '=NC XNC)) ) (LIST 'BEST (LIST 'A2 A2)
(LIST 'B1 B1) (LIST 'B2 B2) (LIST 'C1 C1) (LIST 'C2 C2)) ) RESULT))
                     ((GREATERP OVAM XOVAM)
                        (PRISM:PUSH (LIST (LIST 'RELATION (LIST '(NC =NC) (LIST '#LESSP '=NC (PLUS XNC 1))) ) (LIST 'BEST (LIST 'A
2 XA2) (LIST 'B1 XB1) (LIST 'B2 XB2) (LIST 'C1 XC1) (LIST 'C2 XC2)) ) RESULT))
                     (T (PRISM:PUSH (LIST 'SPAN (LIST 'NC XNC NC)) RESULT))))

               ((GREATERP XNC NC)
                (COND ((GREATERP OVAM XOVAM)
                        (PRISM:PUSH (LIST (LIST 'RELATION (LIST '(NC =NC) (LIST '#GREATER? '=NC NC)) ) (LIST 'BEST (LIST 'A2 XA2)
(LIST 'B1 XB1) (LIST 'B2 XB2) (LIST 'C1 XC1) (LIST 'C2 XC2)) ) RESULT))
                     ((GREATERP XOVAM OVAM)
                        (PRISM:PUSH (LIST (LIST 'RELATION (LIST '(NC =NC) (LIST '#LESSP '=NC (PLUS NC 1))) ) (LIST 'BEST (LIST 'A2
 A2) (LIST 'B1 B1) (LIST 'B2 B2) (LIST 'C1 C1) (LIST 'C2 C2)) ) RESULT))
                     (T (PRISM:PUSH (LIST 'SPAN (LIST 'NC NC XNC)) RESULT))))
         ) )

     (T (COND ((AND OK (NOT XOK)) (PRISM:PUSH (LIST 'IN-SITUATION SIT
                                        SETS 'BETTER-THAN XSETS) RESULT))
              ((AND XOK (NOT OK)) (PRISM:PUSH (LIST 'IN-SITUATION SIT
                                        XSETS 'BETTER-THAN SETS) RESULT)))
        (COND ((GREATERP OVAM XOVAM) (PRISM:PUSH (LIST 'IN-SITUATION SIT
                                        XSETS 'BETTER-THAN SETS) RESULT))
              ((GREATERP XOVAM OVAM) (PRISM:PUSH (LIST 'IN-SITUATION SIT
                                        SETS 'BETTER-THAN XSETS) RESULT)))
     ))

))



(DEXPR CALCULATE-PERFORMANCE (NA NB NC A2 B1 B2 C1 C2 FAR)
       (PROG (RESULT FARU WACQ XACQ YACQ ZACQ WAQ XAQ YAQ ZAQ OVAM ST
             P1 P2 DPCWA DPCXA DPCYA DPCZA PQ1 PQ2 PQ3 PQ4 DPCWQ DPCXQ DPCYQ DPCZQ
             MAX-WACQ MAX-XACQ MAX-YACQ MAX-ZACQ MAX-WQA MAX-XQA MAX-YQA MAX-ZQA)

       (SETQ MAX-WACQ 125) (SETQ MAX-XACQ 125) (SETQ MAX-YACQ 125) (SETQ MAX-ZACQ 125)
       (SETQ MAX-WQA 100)  (SETQ MAX-XQA 100)  (SETQ MAX-YQA 100)  (SETQ MAX-ZQA 100)

       (SETQ ST (FPLUS (FTIMES A2 NA) (FTIMES B1 B2 NB) (FTIMES C1 C2 NC)))

       (COND ((EQP A2 0.1) (SETQ P1 0.1365) (SETQ P2 0.6456))
             ( T           (SETQ P1 0.6443) (SETQ P2 0.9921)) )

       (SETQ DPCWA (FQUOTIENT (FTIMES P1 2.914) ST))
       (SETQ DPCXA (FQUOTIENT (FTIMES P2 2.914) ST))
       (SETQ DPCYA (FQUOTIENT (FTIMES P1 7.258) ST))
       (SETQ DPCZA (FQUOTIENT (FTIMES P2 7.258) ST))
```

```
(COND ((EQP B2 0.1) (SETQ PQ1 0.1365) (SETQ PQ2 0.6456))
      ( T           (SETQ PQ1 0.6443) (SETQ PQ2 0.9921)) )

(COND ((EQP B2 0.15) (SETQ PQ1 0.38417) (SETQ PQ2 0.93148))
      ( T            (SETQ PQ1 0.6443)  (SETQ PQ2 0.9921 )) )

(SETQ DPCWQ (FQUOTIENT (FTIMES PQ1 2.914 B1) ST))
(SETQ DPCXQ (FQUOTIENT (FTIMES PQ2 2.914 B1) ST))

(COND ((EQP C2 0.1) (SETQ PQ3 0.1365) (SETQ PQ4 0.6456))
      ( T           (SETQ PQ3 0.6443) (SETQ PQ4 0.9921)) )

(COND ((EQP C2 0.15) (SETQ PQ3 0.38417) (SETQ PQ4 0.93148))
      ( T            (SETQ PQ3 0.6443 ) (SETQ PQ4 0.9921 )) )

(SETQ DPCYQ (FQUOTIENT (FTIMES PQ3 7.258 C1) ST))
(SETQ DPCZQ (FQUOTIENT (FTIMES PQ4 7.258 C1) ST))

(COND ((GREATERP DPCWA 1) (SETQ WACQ (FNML DPCWA)))
      ( T                 (SETQ WACQ (FNM  DPCWA))) )

(COND ((GREATERP DPCXA 1) (SETQ XACQ (FNML DPCXA)))
      ( T                 (SETQ XACQ (FNM  DPCXA))) )

(COND ((GREATERP DPCYA 1) (SETQ YACQ (FNML DPCYA)))
      ( T                 (SETQ YACQ (FNM  DPCYA))) )

(COND ((GREATERP DPCZA 1) (SETQ ZACQ (FNML DPCZA)))
      ( T                 (SETQ ZACQ (FNM  DPCZA))) )

(COND ((ZEROP NB)         (SETQ WQA 0) (SETQ XQA 0))
      (T (COND ((GREATERP DPCWQ 1) (SETQ WQA (FNML DPCWQ)))
               ( T                 (SETQ WQA (FNM  DPCWQ))) )
         (COND ((GREATERP DPCXQ 1) (SETQ XQA (FNML DPCXQ)))
               ( T                 (SETQ XQA (FNM  DPCXQ))) ) ))

(COND ((ZEROP NC)         (SETQ YQA 0) (SETQ ZQA 0))
      (T  (COND ((GREATERP DPCYQ 1) (SETQ YQA (FNML DPCYQ)))
                ( T                 (SETQ YQA (FNM  DPCYQ))) )
          (COND ((GREATERP DPCZQ 1) (SETQ ZQA (FNML DPCZQ)))
                ( T                 (SETQ ZQA (FNM  DPCZQ))) ) ))

(SETQ FARU 9.996)

(SETQ OVAM (FPLUS FARU (FTIMES 0.25 (FPLUS WACQ XACQ YACQ ZACQ))
                  WQA XQA YQA ZQA))

(PRISM:PUSH (LIST 'FARU FARU) RESULT)
(PRISM:PUSH (LIST 'WACQ WACQ) RESULT)
(PRISM:PUSH (LIST 'XACQ XACQ) RESULT)
(PRISM:PUSH (LIST 'YACQ YACQ) RESULT)
(PRISM:PUSH (LIST 'ZACQ ZACQ) RESULT)
(PRISM:PUSH (LIST 'WQA WQA) RESULT)
(PRISM:PUSH (LIST 'XQA XQA) RESULT)
(PRISM:PUSH (LIST 'YQA YQA) RESULT)
(PRISM:PUSH (LIST 'ZQA ZQA) RESULT)
(PRISM:PUSH (LIST 'OVAM OVAM) RESULT)

(COND ((AND (LESSP WACQ MAX-WACQ)
```

```
                (LESSP XACQ MAX-XACQ)
                (LESSP YACQ MAX-YACQ)
                (LESSP ZACQ MAX-ZACQ)
                (LESSP WQA MAX-WQA)
                (LESSP XQA MAX-XQA)
                (LESSP YQA MAX-YQA)
                (LESSP ZQA MAX-ZQA)) (PRISM:PUSH '(PERFORMANCE-OK) RESULT))
              (T (PRISM:PUSH '(PERFORMANCE-BAD) RESULT)))


        (RETURN RESULT)
    ))

(DEXPR FNM (U) (FTIMES 170 (ANTILOG (FTIMES U -3.52636))))

(DEXPR FNML (U) (FTIMES 8.55 (ANTILOG (FTIMES U -0.5365))))

(DEXPR SET-NEXT-TRY (PARAMS)
       (PROG (RESULT)
       (PRISM:PUSH (LIST 'A2 (CAR PARAMS)) RESULT)
       (PRISM:PUSH (LIST 'B1 (CADR PARAMS)) RESULT)
       (PRISM:PUSH (LIST 'B2 (CADDR PARAMS)) RESULT)
       (PRISM:PUSH (LIST 'C1 (CADDR (CDR PARAMS))) RESULT)
       (PRISM:PUSH (LIST 'C2 (CADDR (CDDR PARAMS))) RESULT)
       (RETURN RESULT) ))


STOP
?1(DEFAULTFONT 1 (GACHA 10) (GACHA 9) (TERMINAL 8))
```

APPENDIX G

Log of a Session Running the Learning Program, DISCRIM

LOG OF A SESSION RUNNING THE LEARNING PROGRAM, DISCRIM

```
Listing of File {FLOPPY}DISC.LOG ,  3-Nov-87 12:06:16 by


11←(CLEAR-ALL)

CLEAR-ALL OLD-CASE-WM:
CLEAR-ALL CANDIDATE-RELATIONS-WM:
CLEAR-ALL LEARN-WM:
CLEAR-ALL TRIES-WM:
CLEAR-ALL BEST-PERFORMANCES-WM:
CLEAR-ALL THIS-CASE-WM:
CLEAR-ALL CASE-WM:
CLEAR-ALL PREVIOUS-CASE-WM:
CLEAR-ALL CURRENT-CASE-WM:
CLEAR-ALL WM:
CLEAR-ALL LEARNING-PM:
CLEAR-ALL INPUT-SITUATION-PM:
CLEAR-ALL TRY-ANYTHING-PM:
CLEAR-ALL PERFORMANCE-CHECK-PM:
CLEAR-ALL SATISFYING-PM:
CLEAR-ALL PM:
NIL
12←(PLOAD DISCRIM)

{DSK}<LISPFILES>DISCRIM.;10

BUILDING  START-PROBLEM
BUILDING  SET-MONITORING-TO-OPTIMIZE
BUILDING  GO-TO-SATISFY
BUILDING  SINGLE-HIGH-SPEED-TARGET
BUILDING  FEW-OTHER-TARGETS
BUILDING  HIGH-SPEED-TARGET
BUILDING  DEFAULT-SIT
BUILDING  OTHER-SIT
BUILDING  FINISHED-PICKING-FIRST-PARAM-SET
BUILDING  ELIMINATE-TRIAL-HIGH
BUILDING  ELIMINATE-TRIAL-OTHER
BUILDING  ELIMINATE-TRIAL
BUILDING  FIRST-SET-IN-SATISFYING
BUILDING  FOR-OPTIMIZING-OR-MONITORING
BUILDING  FOR-LEARNING
BUILDING  CHECK-PERFORMANCE
BUILDING  STORE-CASE-FOR-LEARNING
BUILDING  PERFORMANCE-UNACCEPTABLE
BUILDING  DECIDE-WHAT-CAN-BE-TRIED
BUILDING  REMOVE-SET-TRIED-FIRST
BUILDING  GET-NEW-TRIAL
BUILDING  FINISHED-SATISFYING
BUILDING  FIRST-OPTIMIZING-TRY
BUILDING  CHECK-FOR-OVERALL-BEST-SO-FAR
BUILDING  TRY-ANYTHING-NOT-TRIED
BUILDING  COUNT-POINTS
BUILDING  COMPARE-CASES
ADD-TO TRIES-WM:
(START){DSK}<LISPFILES>DISCRIM.;10
13←(RUN)
```

```
CYCLE 1
-------------------------
FIRING 1
PRODUCTION: START-PROBLEM
MATCHED:
(START)
CLEAR-DM WM:
CLEAR-DM TRIES-WM:
CLEAR-DM CURRENT-CASE-WM:
CLEAR-DM THIS-CASE-WM:
CLEAR-DM BEST-PERFORMANCES-WM:
TYPE A PROBLEM (OR Q TO QUIT):
:: (((NA 28) (NB 3) (NC 5) (CURRENT-BEAM HIGH-SPEED))
...


ADD-TO TRIES-WM:
(FAR 7.2)
(CURRENT-BEAM HIGH-SPEED)
(NC 5)
(NB 3)
(NA 28)
(GOAL SATISFYING)
ADD-TO TRIES-WM:
(FIRST-TRY)
(NOT-ALTERNATIVES-SET)
ADD-TO WM:
(NOT-ELIMINATED)
(NOT-CASE-STORED)
(NOT-PERF-KNOWN)
-------------------------
CYCLE 2
-------------------------
FIRING 2
PRODUCTION: GO-TO-SATISFY
MATCHED:
(FIRST-TRY)
-------------------------
CYCLE 3
-------------------------
FIRING 3
PRODUCTION: HIGH-SPEED-TARGET
MATCHED:
(CURRENT-BEAM HIGH-SPEED)
ADD-TO WM:
(A2 .15)
(B1 2)
(B2 .2)
(C1 2)
(C2 .15)
-------------------------
CYCLE 4
-------------------------
FIRING 4
PRODUCTION: FINISHED-PICKING-FIRST-PARAM-SET
MATCHED:
(A2 .15)
(B1 2)
```

```
(B2 .2)
(C1 2)
(C2 .15)
ADD-TO WM:
(FIRST-SET)
ADD-TO TRIES-WM:
(TRY (.15 2 .2 2 .15))
-------------------------
CYCLE 5
-------------------------
FIRING 5
PRODUCTION: FIRST-SET-IN-SATISFYING
MATCHED:
(GOAL SATISFYING)
(FIRST-SET)
-------------------------
CYCLE 6
-------------------------
FIRING 6
PRODUCTION: CHECK-PERFORMANCE
MATCHED:
(A2 .15)
(NOT-PERF-KNOWN)
(NOT-ELIMINATED)
(B1 2)
(B2 .2)
(C1 2)
(C2 .15)
(NA 28)
(TRY (.15 2 .2 2 .15))
(NB 3)
(NC 5)
(FAR 7.2)
ADD-TO WM:
(PERFORMANCE-OK)
(OVAM 87.71338)
(ZQA 2.987977)
(YQA 9.83334)
(XQA 8.85367)
(WQA 24.947)
(ZACQ 4.884436)
(YACQ 15.57836)
(XACQ 38.79593)
(WACQ 65.12288)
(FARU 9.996)
ADD-TO WM:
(PERF-KNOWN)
DELETE-FROM WM:
(NOT-PERF-KNOWN)
-------------------------
CYCLE 7
-------------------------
FIRING 7
PRODUCTION: STORE-CASE-FOR-LEARNING
MATCHED:
(PERF-KNOWN)
(NOT-CASE-STORED)
(A2 .15)
(B1 2)
```

```
(B2 .2)
(C1 2)
(C2 .15)
(NA 28)
(NB 3)
(NC 5)
(CURRENT-BEAM HIGH-SPEED)
(WACQ 65.12288)
(XACQ 38.79593)
(YACQ 15.57836)
(ZACQ 4.884436)
(WQA 24.947)
(XQA 8.85367)
(YQA 9.83334)
(ZQA 2.987977)
(FARU 9.996)
(DVAM 87.71338)
ADD-TO CASE-WM:
((28 3 5 HIGH-SPEED)
 (.15 2 .2 2 .15)
 (65.12288
  38.79593
  15.57836
  4.884436
  24.947
  8.85367
  9.83334
  2.987977
  9.996
  87.71338))
ADD-TO THIS-CASE-WM:
((28 3 5 HIGH-SPEED)
 (.15 2 .2 2 .15)
 (65.12288
  38.79593
  15.57836
  4.884436
  24.947
  8.85367
  9.83334
  2.987977
  9.996
  87.71338))
ADD-TO CURRENT-CASE-WM:
(CONDITIONS (28 3 5 HIGH-SPEED))
ADD-TO CURRENT-CASE-WM:
(SETTINGS (.15 2 .2 2 .15))
ADD-TO CURRENT-CASE-WM:
(PERFORMANCES
 (65.12288
  38.79593
  15.57836
  4.884436
  24.947
  8.85367
  9.83334
  2.987977
  9.996
  87.71338))
```

```
ADD-TO TRIES-WM:
(TRIED (.15 2 .2 2 .15))
DELETE-FROM TRIES-WM:
(TRY (.15 2 .2 2 .15))
ADD-TO WM:
(CASE-STORED)
DELETE-FROM WM:
(NOT-CASE-STORED)
-------------------------
CYCLE 8
-------------------------
FIRING 8
PRODUCTION: FINISHED-SATISFYING
MATCHED:
(CASE-STORED)
(PERFORMANCE-OK)
(GOAL SATISFYING)*****THE PARAMETERS FOR:
PRINT-FIRSTS TRIES-WM:
(CURRENT-BEAM HIGH-SPEED)
(NA 28)
(NB 3)
(NC 5)**ARE:
PRINT-FIRSTS WM:
(A2 .15)
(B1 2)
(B2 .2)
(C1 2)
(C2 .15)**WITH PERFORMANCE VALUES:
PRINT-FIRSTS WM:
(FARU 9.996)
(WACQ 65.12288)
(XACQ 38.79593)
(YACQ 15.57836)
(ZACQ 4.884436)
(WQA 24.947)
(XQA 8.85367)
(YQA 9.83334)
(ZQA 2.987977)
(OVAM 87.71338)
CLEAR-DM WM:
CLEAR-DM TRIES-WM:
ADD-TO TRIES-WM:
(START)
-------------------------
CYCLE 9
-------------------------
FIRING 9
PRODUCTION: START-PROBLEM
MATCHED:
(START)
CLEAR-DM WM:
CLEAR-DM TRIES-WM:
CLEAR-DM CURRENT-CASE-WM:
CLEAR-DM THIS-CASE-WM:
CLEAR-DM BEST-PERFORMANCES-WM:
TYPE A PROBLEM (OR Q TO QUIT):
:: QQ

EXPLICIT HALT
```

```
9 CYCLES
9 FIRINGS
TOTAL CPU SECONDS = 51.46667
CPU SECONDS PER CYCLE = 5.718517
14←(PMTRACE NAMES)
SET
15←(DMTRACE NONE)
SET


20←(CALL INPUT-SITUATION-PM)
21←(ADD-TO TRIES-WM (START))
NIL
22←(RUN)

FIRING 1
PRODUCTION: GO-TO-SATISFY
NO PRODUCTION ACCEPTABLE
1 CYCLES
1 FIRINGS
TOTAL CPU SECONDS = .9333334
CPU SECONDS PER CYCLE = .9333334
CPU SECONDS PER FIRING = .9333334
CONFLICT SET STATISTICS:
SATISFYING-PM: AVERAGE SIZE = 0.0, MAXIMUM SIZE = 0
INPUT-SITUATION-PM: AVERAGE SIZE = 2.0, MAXIMUM SIZE = 2
FIRING 2
PRODUCTION: START-PROBLEM
CLEAR-DM WM:
CLEAR-DM TRIES-WM:
CLEAR-DM CURRENT-CASE-WM:
CLEAR-DM THIS-CASE-WM:
CLEAR-DM BEST-PERFORMANCES-WM:
TYPE A PROBLEM (OR Q TO QUIT):
:: (((GOAL OPTIMIZING) (NA 28) (NB 3) (NC 5)
(CURRENT-BEAM OTHER))
...



NO PRODUCTION ACCEPTABLE
0 CYCLES
2 FIRINGS
TOTAL CPU SECONDS = 6.0
CPU SECONDS PER CYCLE = UNDEFINED
CPU SECONDS PER FIRING = 3.0NIL
23←(CONTINUE)

FIRING 3
PRODUCTION: OTHER-SIT
FIRING 4
PRODUCTION: FINISHED-PICKING-FIRST-PARAM-SET
FIRING 5
PRODUCTION: FOR-OPTIMIZING-OR-MONITORING
FIRING 6
PRODUCTION: CHECK-PERFORMANCE
FIRING 7
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 8
```

```
PRODUCTION: FIRST-OPTIMIZING-TRY
CLEAR-DM BEST-PERFORMANCES-WM:
*****THE PARAMETERS FOR:
PRINT-FIRSTS TRIES-WM:
(CURRENT-BEAM OTHER)
(NA 28)
(NB 3)
(NC 5)
**ARE:
PRINT-FIRSTS WM:
(A2 .15)
(B1 2)
(B2 .2)
(C1 2)
(C2 .2)
**WITH PERFORMANCE VALUES:
PRINT-FIRSTS WM:
(WACQ 69.48424)
(XACQ 42.86885)
(YACQ 18.30846)
(ZACQ 5.498253)
(WQA 28.40084)
(XQA 10.81022)
(YQA 4.339954)
(ZQA 3.009685)
(FARU 9.996)
(OVAM 90.5968)
FIRING 9
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
FIRING 10
PRODUCTION: CHECK-PERFORMANCE
FIRING 11
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 12
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
FIRING 13
PRODUCTION: CHECK-PERFORMANCE
FIRING 14
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 15
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
FIRING 16
PRODUCTION: CHECK-PERFORMANCE
FIRING 17
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 18
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
FIRING 19
PRODUCTION: CHECK-PERFORMANCE
FIRING 20
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 21
PRODUCTION: CHECK-FOR-OVERALL-BEST-SO-FAR
CLEAR-DM BEST-PERFORMANCES-WM:
*****THE PARAMETERS FOR:
```

```
PRINT-FIRSTS TRIES-WM:
(CURRENT-BEAM OTHER)
(NA 28)
(NB 3)
(NC 5)
**ARE:
PRINT-FIRSTS WM:
(A2 .15)
(B1 2)
(B2 .2)
(C1 2)
(C2 .15)
**WITH PERFORMANCE VALUES:
PRINT-FIRSTS WM:
(WACQ 65.12288)
(XACQ 38.79593)
(YACQ 15.57836)
(ZACQ 4.864436)
(WQA 24.947)
(XQA 8.85367)
(YQA 9.83334)
(ZQA 2.987977)
(FARU 9.996)
(OVAM 87.71338)
FIRING 22
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
FIRING 23
PRODUCTION: CHECK-PERFORMANCE
FIRING 24
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 25
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
FIRING 26
PRODUCTION: CHECK-PERFORMANCE
FIRING 27
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 28
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
FIRING 29
PRODUCTION: CHECK-PERFORMANCE
FIRING 30
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 31
PRODUCTION: CHECK-FOR-OVERALL-BEST-SO-FAR
CLEAR-DM BEST-PERFORMANCES-WM:
*****THE PARAMETERS FOR:
PRINT-FIRSTS TRIES-WM:
(CURRENT-BEAM OTHER)
(NA 28)
(NB 3)
(NC 5)
**ARE:
PRINT-FIRSTS WM:
(A2 .15)
(B1 2)
(B2 .2)
```

```
(C1 1)
(C2 .2)
**WITH PERFORMANCE VALUES:
PRINT-FIRSTS WM:
(WACQ 60.41958)
(XACQ 34.56657)
(YACQ 12.92509)
(ZACQ 4.675394)
(WQA 21.47368)
(XQA 7.028516)
(YQA 12.92509)
(ZQA 4.675394)
(FARU 9.996)
(OVAM 84.24534)
FIRING 32
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
FIRING 33
PRODUCTION: CHECK-PERFORMANCE
FIRING 34
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 35
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
FIRING 36
PRODUCTION: CHECK-PERFORMANCE
FIRING 37
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 38
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
FIRING 39
PRODUCTION: CHECK-PERFORMANCE
FIRING 40
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 41
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
FIRING 42
PRODUCTION: CHECK-PERFORMANCE
FIRING 43
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 44
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
FIRING 45
PRODUCTION: CHECK-PERFORMANCE
FIRING 46
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 47
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
FIRING 48
PRODUCTION: CHECK-PERFORMANCE
FIRING 49
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 50
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
```

```
FIRING 51
PRODUCTION: CHECK-PERFORMANCE
FIRING 52
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 53
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
FIRING 54
PRODUCTION: CHECK-PERFORMANCE
FIRING 55
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 56
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
FIRING 57
PRODUCTION: CHECK-PERFORMANCE
FIRING 58
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 59
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
FIRING 60
PRODUCTION: CHECK-PERFORMANCE
FIRING 61
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 62
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
FIRING 63
PRODUCTION: CHECK-PERFORMANCE
FIRING 64
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 65
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
FIRING 66
PRODUCTION: CHECK-PERFORMANCE
FIRING 67
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 68
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
FIRING 69
PRODUCTION: CHECK-PERFORMANCE
FIRING 70
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 71
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
FIRING 72
PRODUCTION: CHECK-PERFORMANCE
FIRING 73
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 74
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
FIRING 75
PRODUCTION: CHECK-PERFORMANCE
FIRING 76
```

```
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 77
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
FIRING 78
PRODUCTION: CHECK-PERFORMANCE
FIRING 79
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 80
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
FIRING 81
PRODUCTION: CHECK-PERFORMANCE
FIRING 82
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 83
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
FIRING 84
PRODUCTION: CHECK-PERFORMANCE
FIRING 85
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 86
PRODUCTION: TRY-ANYTHING-NOT-TRIED
CLEAR-DM WM:
24←(CALL INPUT-SITUATION-PM)
NIL
25←(CLEAR-DM TRIES-WM)

CLEAR-DM TRIES-WM:NIL
26←(ADD-TO TRIES-WM (START))
NIL
27←(RUN)

FIRING 1
PRODUCTION: START-PROBLEM
CLEAR-DM WM:
CLEAR-DM TRIES-WM:
CLEAR-DM CURRENT-CASE-WM:
CLEAR-DM THIS-CASE-WM:
CLEAR-DM BEST-PERFORMANCES-WM:
TYPE A PROBLEM (OR Q TO QUIT):
:: (((NA 36)))
...



FIRING 2
PRODUCTION: GO-TO-SATISFY
FIRING 3
PRODUCTION: DEFAULT-SIT
FIRING 4
PRODUCTION: FINISHED-PICKING-FIRST-PARAM-SET
FIRING 5
PRODUCTION: FIRST-SET-IN-SATISFYING
FIRING 6
PRODUCTION: CHECK-PERFORMANCE
FIRING 7
PRODUCTION: STORE-CASE-FOR-LEARNING
```

```
FIRING 8
PRODUCTION: FINISHED-SATISFYING*****THE PARAMETERS FOR:
PRINT-FIRSTS TRIES-WM:
(CURRENT-BEAM DEFAULT)
(NA 36)
(NB 0)
(NC 0)**ARE:
PRINT-FIRSTS WM:
(A2 .15)
(B1 2)
(B2 .2)
(C1 1)
(C2 .2)**WITH PERFORMANCE VALUES:
PRINT-FIRSTS WM:
(FARU 9.996)
(WACQ 49.88607)
(XACQ 25.73632)
(YACQ 8.02063)
(ZACQ 4.180924)
(WQA 0)
(XQA 0)
(YQA 0)
(ZQA 0)
(QVAM 31.95199)
CLEAR-DM WM:
CLEAR-DM TRIES-WM:
FIRING 9
PRODUCTION: START-PROBLEM
CLEAR-DM WM:
CLEAR-DM TRIES-WM:
CLEAR-DM CURRENT-CASE-WM:
CLEAR-DM THIS-CASE-WM:
CLEAR-DM BEST-PERFORMANCES-WM:
TYPE A PROBLEM (OR Q TO QUIT):
:: (((NA 32) (NC 4) (CURRENT-BEAM OTHER))
...


FIRING 10
PRODUCTION: GO-TO-SATISFY
FIRING 11
PRODUCTION: OTHER-SIT
FIRING 12
PRODUCTION: FINISHED-PICKING-FIRST-PARAM-SET
FIRING 13
PRODUCTION: FIRST-SET-IN-SATISFYING
FIRING 14
PRODUCTION: CHECK-PERFORMANCE
FIRING 15
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 16
PRODUCTION: FINISHED-SATISFYING*****THE PARAMETERS FOR:
PRINT-FIRSTS TRIES-WM:
(CURRENT-BEAM OTHER)
(NA 32)
(NB 0)
(NC 4)**ARE:
PRINT-FIRSTS WM:
```

```
(A2 .15)
(B1 2)
(B2 .2)
(C1 2)
(C2 .2)**WITH PERFORMANCE VALUES:
PRINT-FIRSTS WM:
(FARU 9.996)
(WACQ 60.41958)
(XACQ 34.56656)
(YACQ 12.92509)
(ZACQ 4.675394)
(WQA 0)
(XQA 0)
(YQA 3.903654)
(ZQA 2.556644)
(OVAM 44.60296)
CLEAR-DM WM:
CLEAR-DM TRIES-WM:
FIRING 17
PRODUCTION: START-PROBLEM
CLEAR-DM WM:
CLEAR-DM TRIES-WM:
CLEAR-DM CURRENT-CASE-WM:
CLEAR-DM THIS-CASE-WM:
CLEAR-DM BEST-PERFORMANCES-WM:
TYPE A PROBLEM (OR 0 TO QUIT):
:: ((NA 32) (NC 4))
...


FIRING 18
PRODUCTION: GO-TO-SATISFY
FIRING 19
PRODUCTION: DEFAULT-SIT
FIRING 20
PRODUCTION: FINISHED-PICKING-FIRST-PARAM-SET
FIRING 21
PRODUCTION: FIRST-SET-IN-SATISFYING
FIRING 22
PRODUCTION: CHECK-PERFORMANCE
FIRING 23
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 24
PRODUCTION: FINISHED-SATISFYING*****THE PARAMETERS FOR:
PRINT-FIRSTS TRIES-WM:
(CURRENT-BEAM DEFAULT)
(NA 32)
(NB 0)
(NC 4)**AFE:
PRINT-FIRSTS WM:
(A2 .15)
(B1 2)
(B2 .2)
(C1 1)
(C2 .2)**WITH PERFORMANCE VALUES:
PRINT-FIRSTS WM:
(FARU 9.996)
(WACQ 52.11901)
```

```
(XACQ 27.53143)
(YACQ 8.944876)
(ZACQ 4.289123)
(WQA 0)
(XQA 0)
(YQA 8.944876)
(ZQA 4.289123)
(OVAM 46.45111)
CLEAR-DM WM:
CLEAR-DM TRIES-WM:
FIRING 25
PRODUCTION: START-PROBLEM
CLEAR-DM WM:
CLEAR-DM TRIES-WM:
CLEAR-DM CURRENT-CASE-WM:
CLEAR-DM THIS-CASE-WM:
CLEAR-DM BEST-PERFORMANCES-WM:
TYPE A PROBLEM (OR Q TO QUIT):
:: (((NA 25) (NB 5) (NC 6))
...



FIRING 26
PRODUCTION: GO-TO-SATISFY
FIRING 27
PRODUCTION: DEFAULT-SIT
FIRING 28
PRODUCTION: FINISHED-PICKING-FIRST-PARAM-SET
FIRING 29
PRODUCTION: FIRST-SET-IN-SATISFYING
FIRING 30
PRODUCTION: CHECK-PERFORMANCE
FIRING 31
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 32
PRODUCTION: FINISHED-SATISFYING*****THE PARAMETERS FOR:
PRINT-FIRST TRIES-WM:
(CURRENT-BEAM DEFAULT)
(NA 25)
(NB 5)
(NC 6)**ARE:
PRINT-FIRST WM:
(A2 .15)
(B1 2)
(B2 .2)
(C1 1)
(C2 .2)**WITH PERFORMANCE VALUES:
PRINT-FIRST WM:
(PARU 9.996)
(WACQ 65.57398)
(XACQ 39.2105)
(YACQ 15.84852)
(ZACQ 4.904149)
(WQA 25.2938)
(XQA 9.0439)
(YQA 15.84852)
(ZQA 4.904149)
(OVAM 96.47066)
```

```
CLEAR-DM WM:
CLEAR-DM TRIES-WM:
FIRING 33
PRODUCTION: START-PROBLEM
CLEAR-DM WM:
CLEAR-DM TRIES-WM:
CLEAR-DM CURRENT-CASE-WM:
CLEAR-DM THIS-CASE-WM:
CLEAR-DM BEST-PERFORMANCES-WM:
TYPE A PROBLEM (OR Q TO QUIT):
:: (((NA 33) (NB 1) (NC 2) (CURRENT-BEAM HIGH-SPEED))
...


FIRING 34
PRODUCTION: GO-TO-SATISFY
FIRING 35
PRODUCTION: HIGH-SPEED-TARGET
FIRING 36
PRODUCTION: FINISHED-PICKING-FIRST-PARAM-SET
FIRING 37
PRODUCTION: FIRST-SET-IN-SATISFYING
FIRING 38
PRODUCTION: CHECK-PERFORMANCE
FIRING 39
PRODUCTION: STORE-CASE-FOR-LEARNING
FIRING 40
PRODUCTION: FINISHED-SATISFYING*****THE PARAMETERS FOR:
PRINT-FIRST6 TRIES-WM:
(CURRENT-BEAM HIGH-SPEED)
(NA 33)
(NB 1)
(NC 2)**ASE:
PRINT-FIRST6 WM:
(A2 .15)
(B1 2)
(B2 .2)
(C1 2)
(C2 .15)**WITH PERFORMANCE VALUES:
PRINT-FIRST6 WM:
(FARU 9.995)
(WAC0 55.87264)
(XAC0 30.64332)
(YAC0 10.6365?)
(ZAC0 4.466753)
(W0A 18.76324)
(X0A 5.523605)
(Y0A 6.238458)
(Z0A 2.506251)
(0VAM 68.05209)
CLEAR-DM WM:
CLEAR-DM TRIES-WM:
FIRING 41
PRODUCTION: START-PROBLEM
CLEAR-DM WM:
CLEAR-DM TRIES-WM:
CLEAR-DM CURRENT-CASE-WM:
CLEAR-DM THIS-CASE-WM:
```

```
CLEAR-DM BEST-PERFORMANCES-WM:
TYPE A PROBLEM (OR Q TO QUIT):
:: (((GOAL LEARNING) (NA 32) (NB 1) (NC 3)
(CURRENT-BEAM OTHER))
...



FIRING 42
PRODUCTION: GO-TO-SATISFY
FIRING 43
PRODUCTION: OTHER-SIT
FIRING 44
PRODUCTION: FINISHED-PICKING-FIRST-PARAM-SET
FIRING 45
PRODUCTION: FOR-LEARNING
FIRING 46
PRODUCTION: CHECK-PERFORMANCE
FIRING 47
PRODUCTION: STORE-CASE-FOR-LEARNING
NO PRODUCTION ACCEPTABLE
33 CYCLES
47 FIRINGS
TOTAL CPU SECONDS = 169.1333
CPU SECONDS PER CYCLE = 5.094949

29←(PMTRACE FULL)
SET
30←(DMTRACE FULL)
SET
31←(CONTINUE)

-------------------------
CYCLE 34
PRINT-CONTENTS CASE-WM:
((32 1 3 OTHER)
 (.15 2 .2 2 .2)
 (60.41958
  34.56657
  12.92509
  4.673394
  21.47368
  7.028516
  3.903664
  2.556645
  9.996
  73.10517))
((33 1 2 HIGH-SPEED)
 (.15 2 .2 2 .15)
 (59.97264
  30.64332
  10.63659
  4.466753
  16.36324
  5.523605
  6.238458
  2.526251
  9.996
  68.05239))
```

```
((25 5 6 DEFAULT)
 (.15 2 .2 1 .2)
 (65.57398
  39.2105
  15.84852
  4.904149
  25.2938
  9.0439
  15.84852
  4.904149
  9.996
  96.47066))
((32 0 4 DEFAULT)
 (.15 2 .2 1 .2)
 (52.11901 27.53143 8.944876 4.289123 0 0 8.944876 4.289123
 9.996 46.45111))
((32 0 4 OTHER)
 (.15 2 .2 2 .2)
 (60.41958 34.56656 12.92509 4.675394 0 0 3.903664 2.556644
 9.996 44.60296))
((36 0 0 DEFAULT)
 (.15 2 .2 1 .2)
 (49.68607 25.73632 8.02063 4.180924 0 0 0 0 9.996 31.95199
))
((28 3 5 OTHER)
 (.1 2 .2 1 .15)
 (126.5331
  42.06235
  81.47518
  5.244219
  10.46601
  4.449867
  21.48127
  3.984175
  9.996
  114.176))
((28 3 5 OTHER)
 (.15 2 .2 1 .2)
 (60.41956
  34.56657
  12.92509
  4.675394
  21.47363
  7.026516
  12.92509
  4.675394
  9.996
  84.24534))
((28 3 5 OTHER)
 (.15 2 .15 2 .15)
 (62.34361
  36.27602
  13.97472
  4.7617
  51.39483
  9.349092
  8.679526
  2.848546
  9.996
```

```
  111.566))
((28 3 5 OTHER)
 (.15 2 .15 2 .2)
 (66.90707
  40.44466
  16.66322
  4.96208
  55.91216
  11.46778
  4.217375
  2.879794
  9.996
  116.7174))
((28 3 5 OTHER)
 (.15 2 .2 2 .15)
 (65.12288
  38.79593
  15.57836
  4.884436
  24.947
  8.85367
  9.83334
  2.987977
  9.996
  87.71338))
((28 3 5 OTHER)
 (.2 1 .2 1 .2)
 (67.77925
  41.25932
  17.20951
  4.999721
  67.77925
  41.25932
  17.20951
  4.999721
  9.996
  174.0557))
((28 3 5 OTHER)
 (.2 2 .2 1 .2)
 (72.74717
  46.00702
  20.52497
  6.556113
  31.1303
  12.45086
  20.52497
  6.556113
  9.996
  117.1171))
((28 3 5 OTHER)
 (.2 2 .2 2 .2)
 (80.11359
  53.27366
  26.09974
  9.490866
  37.75404
  16.75734
  4.834303
  3.552321
```

```
  9.996
  115.1644))
((28 3 5 OTHER)
 (.15 2 .2 2 .2)
 (69.48484
  42.86885
  18.30846
  5.498253
  28.40084
  10.81022
  4.339954
  3.009685
  9.996
  90.5968))
((28 3 5 HIGH-SPEED)
 (.15 2 .2 2 .15)
 (65.12289
  38.79593
  15.57836
  4.984436
  24.947
  8.85367
  9.83334
  2.987977
  9.996
  87.71338))
PRINT-CONTENTS CURRENT-CASE-WM:
(PERFORMANCES
 (60.41958
  34.56657
  12.92509
  4.675394
  21.47363
  7.028516
  3.903664
  2.556645
  9.996
  73.10517))
(SETTINGS (.15 2 .2 2 .2))
(CONDITIONS (32 1 3 OTHER))
PRINT-CONTENTS WM:
(CASE-STORED)
(PERF-KNOWN)
(FARU 9.996)
(WACQ 60.41958)
(XACQ 34.56657)
(YACQ 12.92509)
(ZACQ 4.675394)
(WQA 21.47363)
(XQA 7.028516)
(YQA 3.903664)
(ZQA 2.556645)
(QVAM 73.10517)
(PERFORMANCE-OK)
(FIRST-SET)
(C2 .2)
(C1 2)
(B2 .2)
(B1 2)
```

(A2 .15)
(NOT-ELIMINATED)